Trabajo de Fin de grado

CleanQuest. Desarrollo de un simulador para fomentar la colaboración infantil en las tareas domésticas

Grado en Ingeniería Informática

Autor: Pablo Sánchez Ordaz

Tutor: María Carmen Fernández Panadero

Director: Jorge Ruiz Magaña

Leganés, enero del 2015

Universidad Carlos III de Madrid

Agradecimientos

Quería dar las gracias por todo el apoyo que he recibido durante la realización de este TFG, en primer lugar a mi familia, especialmente a mis padres y hermano, ya que sin ellos, todo esto no hubiera sido posible.

También agradecer la ayuda y el apoyo prestado por todos mis compañeros de la universidad, en especial Alberto, Diego y Jorge. Gracias a ellos he podido aguantar estos duros años de la carrera.

Por último agradecer a Jorge y M. Carmen, mis tutores del TFG. Sin ellos no hubiera sido posible realizar un TFG como éste, que me ha acercado un poco más a mi sueño, convertirme en desarrollador de videojuegos y me permitirá terminar la carrera a lo grande.

Índice general

Índice de figuras

Índice de tablas

Resumen

Este documento recoge la memoria del trabajo de fin de grado (TFG) con el título "*CleanQuest*. Desarrollo de un simulador para fomentar la colaboración infantil en las tareas domésticas". Consiste en el desarrollo de una herramienta, basada en técnicas de videojuegos, que permita resolver el problema de desobediencia o indiferencia infantil en la realización de las tareas domésticas de limpieza del hogar.

Para ello, primeramente se realiza un pequeño estudio del **problema** propuesto a modo de introducción general para este TFG. En esta introducción se define el problema, se identifica su alcance y se marcan los primeros límites para el desarrollo. Igualmente, a partir de este planteamiento se sintetizan una serie de objetivos a cumplir a través del desarrollo.

Para hallar la **solución** óptima al problema planteado, inicialmente se realiza un análisis del estado del arte. En él se analizan las maneras de abordar el problema en la actualidad, centrándose sobre todo en los productos de software que lo solucionan o ayudan a solucionarlo, junto con los sistemas y kits de desarrollo de su entorno. A través de este análisis se establece la solución óptima como el videojuego educativo *CleanQuest*, con un género y unas características base determinados. En este caso el género del videojuego se establece como una compilación de mini-juegos, con funciones de realidad aumentada. El sistema que lo sustenta por su parte se establece como dispositivos móviles con sistema operativo *Android*, siendo *Unity3d* el kit de herramientas de desarrollo de videojuegos elegido.

A continuación, ya con la solución establecida como un videojuego, se pasa a su desarrollo. Inicialmente se definen en su totalidad todos los aspectos del videojuego a partir de sus características base (elementos, temática, mecanismos, lenguaje, etc.) mediante la concepción y el diseño, dejándolo listo para su implementación. Ésta se realiza utilizando el kit de desarrollo elegido y las herramientas secundarias seleccionadas en el diseño, dando lugar a un prototipo de software de alto nivel de fidelidad.

Una vez finalizado el videojuego se procede a confirmar sus **beneficios** como herramienta para la solución. Para ello se realiza una extensa fase de pruebas con el prototipo de software. Las pruebas definitivas son realizadas con un conjunto de niños que encajan en el rango de edad adecuado para comprobar que mediante el uso del videojuego se soluciona el problema definido inicialmente. Los resultados de las pruebas reflejan que efectivamente el videojuego resuelve el problema y cumple su propósito. Igualmente se plantea el modelo de negocio que debería tener el videojuego, atendiendo a las características finales del prototipo software, para confirmar su viabilidad comercial. En este caso una distribución gratuita en *Google Play Store*, con posibilidad de añadir nuevo contenido descargable con micro pagos.

Estructura del documento

En este apartado se describe la estructura de este documento en los siguientes apartados. Se consideran sólo los apartados de primer nivel, describiendo de forma breve el contenido de cada uno:

- **1. Introducción.** Definición del problema planteado en la actualidad, indicando su solución teórica, el videojuego, junto con los beneficios de la misma. A su vez, se establecen los objetivos a cumplir mediante el desarrollo del videojuego, la motivación para cumplirlos y los medios y metodologías generales empleados en el desarrollo. Por último se introduce a las fases en las que se estructura el proceso de desarrollo.

- **2. Estado del arte.** Análisis del estado del arte relativo a la resolución del problema planteado. Se analizan una serie de productos de software, existentes en la actualidad, que comparten objetivos de la solución, con el fin de establecer el género y las características base del videojuego a desarrollar, y se considera un posible modelo de negocio para el mismo. Igualmente se analizan los sistemas actuales y los kits de desarrollo de videojuegos actuales más adecuados para albergar y desarrollar respectivamente un videojuego con las características base establecidas.

- **3. Desarrollo.** Recoge las partes más relevantes del proceso de desarrollo del videojuego, pasando por la preproducción (concepción, diseño y planificación), producción (implementación) y postproducción.

- **4. Pruebas.** Se exponen las pruebas de evaluación unitarias, de integración, de sistema y de validación de producto, junto con sus resultados, con el fin de demostrar el buen funcionamiento del videojuego y validar su propósito.

- **5. Conclusiones y líneas futuras.** Conclusiones de todo el proceso de desarrollo, donde se volverán a listar los objetivos de la introducción y se especificará como se han cumplido uno a uno, además de una valoración de la experiencia personal conseguida con la realización del TFG. Igualmente se consideran las líneas futuras del TFG.

1.Introducción

En este apartado se presenta el problema del que parte todo este TFG y se establecen los primeros límites del desarrollo. Se introduce la definición del problema, la solución propuesta del mismo y los beneficios que conllevaría al utilizarse. Así mismo, se establecen los objetivos concretos para alcanzar esta solución, la motivación necesaria y los medios y metodologías empleados para su desarrollo. Por último se describen a grandes rasgos las fases de las que está compuesto el proceso para desarrollar la solución.

1.1 Problema, solución y beneficios

1.1.1 Problema

El problema en el que se basa este TFG es **la desobediencia o indiferencia infantil en la colaboración en las tareas de limpieza del hogar**.

Los niños de 6 a 9 años de edad ya están en disposición de colaborar en determinadas tareas de limpieza y orden en el hogar [1]. En la actualidad una gran cantidad de estos niños no colaboran en este tipo de tareas [2] [3], ya sea por desobediencia directa, pasotismo u otros factores externos, como la situación familiar o una indisposición física o mental.

Esta tendencia se da en todo el mundo, pero es más prominente en países desarrollados, como España. Igualmente acentúa la desigualdad social entre sexos, ya que los niños están condicionados socialmente para ayudar menos que las niñas en este tipo de tareas [4].

Este problema continuado, sin tratarse, puede provocar disfunciones familiares, deficiencias higiénicas (problemas de salud) y un mal desarrollo psicológico en los niños, resultando en adolescentes problemáticos o malacostumbrados.

1.1.2 Solución

Múltiples pedagogos coinciden en que una de las mejores formas de modificar el comportamiento de los niños de cara al problema propuesto, es mediante el uso de incentivos [5] [6] [7] y fomentando positivamente las tareas de limpieza, como si de un juego o desafío se tratasen [8] [9]. El medio y la forma de proporcionar los incentivos deben de ser afines al rango de edad de los niños a los que va dirigido. El rango propuesto para este trabajo comprende a niños de entre 6 y 9 años. No es trivial, ya que en este rango de edad los niños suelen ser más susceptibles a ver las tareas domésticas como una actividad lúdica en vez de una obligación [2] [8].

Recientes trabajos demuestran que los medios que exponen historias o situaciones, como los cuentos, las películas y los videojuegos, son los adecuados para modificar la conducta en este tipo de niños [10] [11].

Para este trabajo se propone un **videojuego** como **medio o herramienta**, cuyo contenido deberá **incentivar** a sus jugadores para colaborar en las tareas de limpieza del hogar y **fomentarlas** de forma positiva como un juego o desafío a completar.

1.1.3 Beneficios

Por supuesto, el beneficio directo de la utilización del videojuego, es un **aumento neto de la colaboración en las tareas de limpieza del hogar**, igualmente pudiendo fomentar la colaboración en tareas domésticas de otra índole.

Adicionalmente, al tratarse de un videojuego con tareas que requieren reflejos y coordinación, el producto final ayudará a aumentar las capacidades coordinativas del jugador (orientación, equilibrio, ritmo, etc.) y la toma rápida de decisiones entre otras [12]. Estas capacidades todavía están en desarrollo en niños de la edad a los que va dirigido el producto, por lo que este apoyo extra sería beneficioso [13] [14].

1.2 Objetivos

Para poder alcanzar la solución al problema planteada en el apartado anterior, es necesario establecer unos **objetivos principales y secundarios**. Éstos se establecen a partir de la propia solución expuesta y se utilizarán como guías y metas globales en todo el proceso del desarrollo.

Como objetivos **principales** se determinan:

- Desarrollar un videojuego que pueda ser utilizado como herramienta para resolver el problema de la desobediencia o indiferencia infantil ante la colaboración en las tareas de limpieza del hogar de una forma óptima (a corto y a largo plazo).

- Seleccionar los medios y metodologías óptimos y el sistema adecuado que soporte el videojuego, para el desarrollo.

- Crear un plan de distribución adecuado para el videojuego, permitiendo que por un lado pueda cumplir su propósito principal (resolver el problema) sin inconvenientes y por otro su monetización, explotando el creciente mercado de los videojuegos educativos.

Como objetivos **secundarios** se determinan:

- Adquirir experiencia como desarrollador de videojuegos de cara al mundo laboral.

Al finalizar todo el proceso de desarrollo, como conclusiones, se volverán a listar todos estos objetivos, especificando uno a uno como han sido satisfechos.

1.3 Motivación

La motivación necesaria, combustible para poder cumplir los objetivos establecidos en el apartado anterior, puede resumirse en los siguientes puntos. Estos están divididos en motivación de carácter general o profesional y en motivación de carácter personal.

Como puntos de carácter general o profesional, se pueden considerar:

- **Necesidad de solucionar o paliar problemas sociales de ésta índole.** El videojuego, fruto del desarrollo, puede ser utilizado como herramienta para solucionar casos individuales. Contribuyendo así a paliar el problema de la desobediencia o indiferencia infantil ante la colaboración en las tareas de limpieza del hogar en la sociedad. Esta necesidad se acentúa con la actual crisis a la que está sometida la sociedad, que en algunos casos obliga a que los niños dentro de las familias tengan que colaborar en casa, para que sus padres puedan continuar con sus extenuantes horarios laborales [15]. El videojuego ayudaría a estos niños a ver esta obligación como un juego o desafío en su lugar.

- **Necesidad de aprovechar el creciente mercado de los videojuegos sociales y educativos.** Tras el boom de los videojuegos en el mercado casual iniciado por *Nintendo* en el 2012, el éxito de este tipo de videojuegos no ha parado de crecer hasta la actualidad [16] [17]. Los videojuegos sociales y educativos forman parte de este nuevo mercado casual y son considerados por las empresas como una apuesta segura, sobre todo si se ofrecen igualmente en sistemas casuales, como los dispositivos móviles. El videojuego a desarrollar por lo tanto, debido a su condición casual, tiene un potencial de rentabilidad muy alto. Así mismo, si el videojuego alcanza el éxito, se contribuiría a su vez a paliar la necesidad del punto anterior.

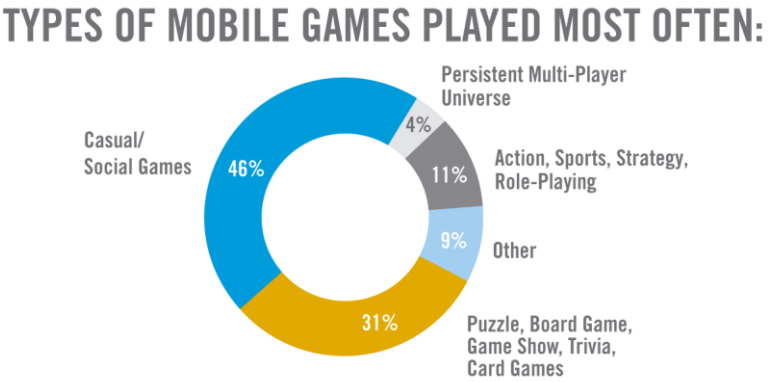


Figura 1.1: Tipos de juegos para móviles más jugados en EEUU (2013)[1]

Como puntos de carácter personal, se pueden considerar:

[1] Fuente: <http://www.theesa.com/wp-content/uploads/2014/10/ESA_EF_2014.pdf>.

- **Adquisición de experiencia personal en el desarrollo de videojuegos.** Los videojuegos son un producto de software extremadamente multidisciplinar, ya que requieren conocimientos de informática, diseño gráfico y artístico, composición, etc. El desarrollo de este videojuego aportaría gran cantidad de experiencia de cara a una futura carrera profesional, con la posibilidad inmediata de empezar a desarrollar otros videojuegos como desarrollador *free-lance*, una vez terminado éste.

- **Reto personal.** "*Desde pequeño he mantenido una gran pasión por los videojuegos y siempre he querido hacer uno. Ésa fue una de las razones principales por las que elegí esta carrera. A lo largo de mi vida, en múltiples ocasiones llegué a crear videojuegos muy básicos y sencillos, pero nada tan ambicioso como éste. Lo considero como el primer videojuego en condiciones que voy a realizar y el culmen de toda la experiencia como estudiante que poseo hasta la fecha*".

1.4 Metodología y medios empleados

1.4.1 Metodología

El desarrollo de videojuegos, a diferencia del desarrollo de software convencional, no tiene definida una metodología estándar. Esto es debido a su condición de producto de software de entretenimiento extremadamente multidisciplinar (fruto del trabajo de programadores, diseñadores gráficos, compositores, etc.) y al actual estado creciente y mutable de su industria, que apenas dispone de convenios de desarrollo lo suficientemente estandarizados [18].

Por este motivo no es factible utilizar únicamente las metodologías típicas usadas en el desarrollo de software convencional (ej. Modelo en cascada). En su lugar suelen utilizarse otras metodologías más flexibles, rápidas y cercanas al usuario, como por ejemplo combinaciones de métodos de desarrollo ágiles o procesos de software personal (PSP).

Para este trabajo, se utilizará una metodología personalizada, resultado de la combinación de varias. Más concretamente, una combinación de las metodologías ágiles [19] de **desarrollo de software adaptativo** [20] **e iterativo** [21] para el desarrollo de la jugabilidad[2] del videojuego, mientras que para creación de *assets* [3] se utilizará una metodología cercana al **modelo en cascada** [22].

La justificación de esta elección es que en la creación de *assets*, los requisitos y especificaciones de los mismos son muy claros desde las fases de concepción y diseño en la preproducción, con poca probabilidad de cambio durante la producción (perfecto para el modelo en cascada). La jugabilidad del videojuego, por el contrario, es más susceptible a cambios incluso durante el proceso de producción (ej. En las pruebas, resultado de la retroalimentación de los usuarios), por lo que es más adecuado abordarla con un modelo de desarrollo ágil adaptativo e iterativo.

[2] *Interacción directa del jugador con el videojuego; combinación de mecanismos y funcionalidades del mismo [23].*

[3] *En el contexto de los videojuegos, elementos individuales que los componen (ej. Personajes, texturas, canciones, etc.).*

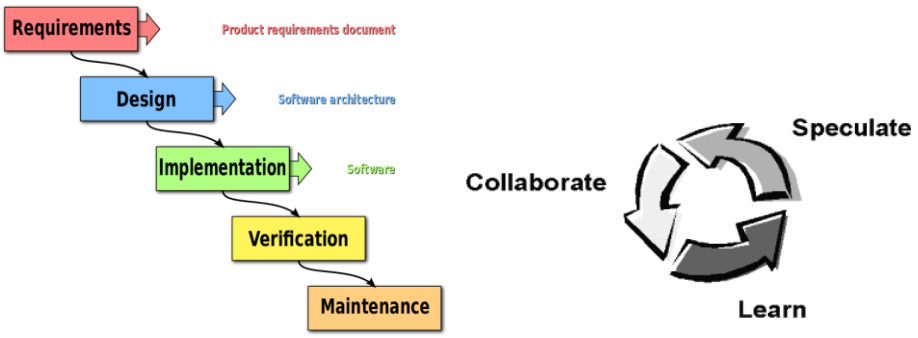


Figura 1.2: Flujo de las metodologías en cascada[4] y de desarrollo adaptativo[5]

1.4.2 Medios

A continuación se presenta una lista con todos los medios finales, empleados en el desarrollo en el desarrollo del videojuego objeto de este TFG. La justificación para el uso de cada uno de ellos se encuentra en los apartados posteriores, pertenecientes a la fase de preproducción. Más concretamente en el análisis del estado del arte y en los sub-apartados de concepción y diseño dentro del propio apartado de preproducción.

Los medios de tipo hardware empleados para el desarrollo son:

- Ordenador de sobremesa *Intel Core 2 Quad Q6600 @ 2.40GHz*
- Ordenador portátil *ASUS Notebook N61Jq Series Intel Core i7 720QM @ 1.60GHz*
- Teléfono móvil (*Smartphone*) *Sony Xperia Neo V MT11i*
- Teléfono móvil (*Smartphone*) *Sony Xperia Z1 C6903*
- Teléfono móvil (*Smartphone*) *Huawei P6-U06*
- Cámara de video digital *Samsung PL21*

Los medios de tipo software empleados para el desarrollo son:

- Sistema Operativo *Windows 7 Ultimate 64 bits*
- Sistema Operativo *Windows 7 Home Premium 64 bits SP1*
- Sistema Operativo *Android 2.3.4 Gingerbread*
- Sistema Operativo *Android 4.4.4 Kit Kat*
- Sistema Operativo *Android 4.2.2 Jelly Bean*
- Kit de desarrollo de videojuegos *Unity 4* Versión *4.5.0f6*
- Entorno de desarrollo *MonoDevelop-Unity* Versión *4.0.1*
- Programa de modelado 3d *Sculptris Alpha 6*
- Programa de modelado y composición 3d *Blender* Versión *2.71*
- Editor de gráficos rasterizados *Adobe Photoshop CS5 Extended*
- Editor de efectos visuales y *composición Adobe After Effects CS5*
- Editor de audio *Audacity 2.0.3*
- Capturador de pantalla *Fraps 3.5.99*

[4] Fuente: <http://en.wikipedia.org/wiki/Waterfall_model>.
[5] Fuente: <http://www.adaptivesd.com/articles/messy.htm>.

1.5 Fases del proceso

Para establecer las fases que van a formar el proceso de elaboración de la solución al problema planteado, hay que tener en cuenta 2 aspectos: El proceso de desarrollo de videojuegos habitual y la investigación del problema en sí. Esto es debido a que la solución propuesta, el videojuego a desarrollar, a diferencia de un videojuego convencional, tiene un **propósito determinado para resolver un problema de índole social**. Por este motivo, con el fin de establecer las fases del proceso, es necesario considerar el proceso de desarrollo habitual de un videojuego y añadirle tareas ajenas o modificar las existentes, para poder incluir aspectos como la **investigación del problema**, **su resolución específica por medio del videojuego** y la **confirmación de su eficacia**.

El proceso de desarrollo habitual en el desarrollo de videojuegos puede parecer, a grandes rasgos, similar al de cualquier producto software; pero debido a su condición de producto de software de entretenimiento, como se ha comentado en el apartado anterior, hay que considerar fases de desarrollo alternativas. Éstas son más parecidas al desarrollo de una película de cine, que al de un producto de software al uso.

De igual forma, este proceso de desarrollo es modular, con un flujo de trabajo complejo, donde muchos de esos módulos de desarrollo se realizan en paralelo. Además, cada módulo suele ser realizado por un grupo de personas distinto, especializado en la disciplina que requiera dicho módulo (programadores, modeladores, animadores, etc.). Como para este videojuego sólo se cuenta con una sola persona para el desarrollo, es necesario adaptar el proceso para hacerlo simple y flexible. Para ello, se contrastan diversos modelos de procesos de desarrollo de videojuegos relevantes. La mayoría de ellos poseen fases similares, variando en algunos casos el orden de las mismas. Aun así es sencillo identificar el patrón de fases principales: Preproducción, producción y postproducción [24] [25].

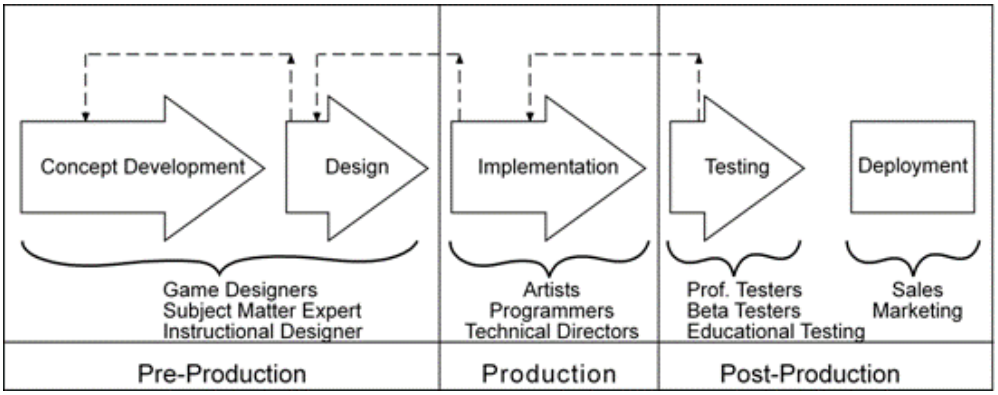


Figura 1.3: Ejemplo de proceso de desarrollo de videojuegos [6]

Atendiendo a todo lo anterior, se plantean un nuevo conjunto de fases para este TFG, basadas en una simplificación de las fases que comúnmente se utilizan en el

[6] Fuente: <http://www.ieeetclt.org/issues/january2010/index.htm>

desarrollo de videojuegos profesional e incorporando los aspectos sobre la resolución del problema planteado:

- **Fase 1: Investigación del problema.** Fase de investigación del problema, su solución y sus beneficios. Esta fase comienza por la **introducción al problema**, donde se define el problema planteado, se identifica su alcance y se marcan los primeros límites para el desarrollo. Seguidamente se realiza un **análisis del estado del arte**, donde se analizan las formas que existen en la actualidad para solucionar el problema, centrándose en los productos de software que se utilizan como herramientas para solucionarlo. A través de este análisis se determina el género del videojuego objetivo (la solución), sus características base, un posible modelo de negocio para el mismo, el sistema que lo soporte y el kit de desarrollo de videojuegos adecuado para su desarrollo.

- **Fase 2: Preproducción.** Fase de conceptualización y diseño. Ya con el videojuego objetivo establecido (herramienta para solucionar el problema), se procede a su preparación de forma que quede listo para su implementación. En esta fase, tienen lugar las sub-fases de **concepción**, **diseño** y **planificación**, donde se establecen todos los conceptos y decisiones de diseño necesarios para definir el videojuego de manera que solucione el problema planteado de la forma establecida en la fase anterior.

- **Fase 3: Producción.** Fase de implementación. En esta fase se realiza la **implementación** del videojuego en base a todas las decisiones tomadas durante la fase anterior. Se comenta el proceso de implementación por encima, destacando cualquier hito o cambios relevantes ocurridos en el mismo. La implementación finaliza con un **prototipo software de alto nivel de fidelidad**, fruto de la misma.

- **Fase 4: Pruebas.** Fase de validación y testeo. Extensa fase donde se realizan las pruebas necesarias para validar el prototipo de software de alto nivel de fidelidad, comprobando que su funcionalidad es la adecuada y que efectivamente cumple su propósito: Resolver el problema. Se realizan **pruebas unitarias**, **de integración**, **de sistema** y **de validación de producto**.

- **Fase 5: Postproducción.** Fase de despliegue y planificación. Una vez validado el prototipo de software, en esta fase se establece el **modelo de negocio** definitivo para su despliegue y distribución, atendiendo a toda la información y decisiones tomadas durante la preproducción y a las características finales del prototipo de software. Igualmente se esboza un plan de **gestión de parches y actualizaciones** futuras para el videojuego.

2.Estado del arte

En este apartado se realizan un conjunto de análisis del estado del arte relativos a la resolución del problema planteado inicialmente: La desobediencia o indiferencia infantil ante la colaboración en las tareas de limpieza del hogar. Comienza con una breve introducción, presentando el estado del arte general de los artefactos y métodos actuales que solucionan o ayudan a solucionar el problema.

A continuación se realiza un extenso análisis del estado del arte de los productos de software que comparten alguno o algunos de los objetivos de la solución. Este análisis se centra en las características de los videojuegos u otras aplicaciones relevantes, que existan en la actualidad y solucionen o ayuden a solucionar el mismo problema. A través de este análisis se determina el género y las características base del videojuego, fruto del desarrollo, y un posible modelo de negocio para el mismo.

Para finalizar, se hace un último análisis del estado del arte sobre los sistemas y los kits de desarrollo, utilizados para soportar y desarrollar videojuegos con las mismas características. El objetivo es seleccionar el sistema y el kit de desarrollo más adecuados para el desarrollo.

2.1 Introducción al estado del arte relativo a la solución del problema planteado

La solución teórica global al problema de la desobediencia o indiferencia infantil ante la colaboración en las tareas de limpieza del hogar es extremadamente compleja y abarca múltiples campos profesionales. Campos como la psicología infantil, sociología, estudios de género, etc [2]. Es por ello que, incluso en la actualidad, apenas existen artefactos o métodos verdaderamente efectivos que solucionen el problema.

Como se comentó en el apartado de "Problema, solución y beneficios" (apartado 1.1), una de las mejores y más comunes formas de modificar el comportamiento de los niños de cara al problema planteado, es mediante el **uso de incentivos** y **fomentando la visión positiva de las tareas de limpieza**. Para este problema en particular, éstos incentivos son algún tipo de premio, no necesariamente material, que se otorga al niño/a al terminar alguna tarea de limpieza.

Teniendo esto en cuenta, en la actualidad, muchas familias recurren a artefactos y métodos con base en algún tipo de incentivo para solucionar o paliar casos individuales. Se utilizan tanto artefactos y métodos tradicionales como nuevos y más tecnológicos.

Entre los artefactos y métodos tradicionales destacan las tablas de tareas. Se trata de tablas de celdas vacías que reflejan las tareas domésticas que el niño debe realizar durante un determinado tiempo. Cuando el niño/a termina una tarea, lo indica en la tabla, marcando la celda vacía correspondiente a esa tarea.

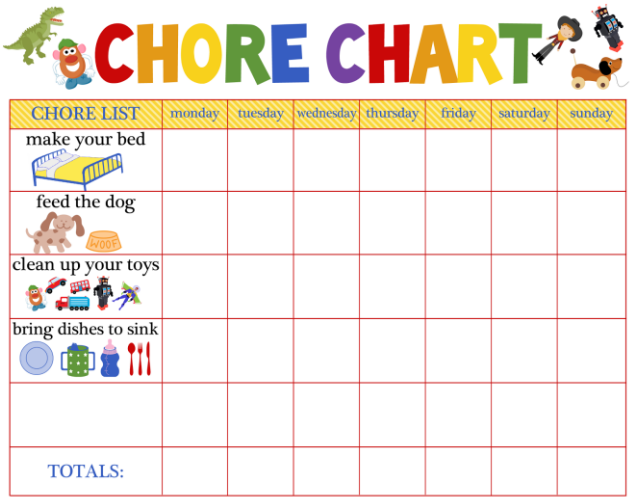


Figura 2.1: Ejemplo de tabla de tareas[7]

[7] Fuente: <http://parentsfourchildren.com/children-doing-chores-are-building-strong-foundations/>.

El incentivo de este método se basa en que el niño, al completar la tabla o parte de ella, recibe algún tipo de premio. Mediante su uso, si se usa de forma adecuada, se garantiza que el niño/a colaborará en las tareas del hogar (incluyendo las de limpieza).

El principal inconveniente que presenta este método es que gran parte de su efectividad se basa en la calidad del premio. En otras palabras, a parte de la calidad del premio y la característica visual del progreso en la tabla, no presenta ningún otro incentivo o atractivo. Esto puede no ser suficiente para el niño/a, teniendo que intervenir el padre, obligándolo de alguna manera a participar.

Igualmente, en la actualidad, existen muchos otros métodos o artefactos tradicionales, menos destacables o personalizados, con el mismo objetivo; como las tarjetas de tareas, juegos de selección aleatoria de tareas, concienciación de tratar las tareas como un juego, programas de televisión, etc. La mayoría de éstos son muy similares o parten de los mismos principios que la tabla de tareas, compartiendo ventajas e inconvenientes. Cabe señalar a los juegos de selección aleatoria de tareas, que añaden un punto de aleatoriedad y sorpresa como un incentivo más.

Figura 2.2: Ejemplos de juegos de selección aleatoria de tareas[8]

Como artefactos o métodos tradicionales que fomentan la visión positiva de las tareas de limpieza se podrían considerar juguetes u otro tipo de juegos cuyo propósito es la simulación de la limpieza de la casa, como por ejemplo un kit de limpieza infantil para niños. El problema de estos artefactos es que, a diferencia de los comentados anteriormente, mientras que hacen más atractivas estas tareas a los ojos de los niños, no garantizan que los niños verdaderamente las realicen, simplemente fomentan su realización.

[8] Fuente: <http://www.craftinterrupted.com/2011/05/creative-chore-plans-for-kids.html>.

Figura 2.3: Ejemplo de juguete que fomenta la limpieza en casa[9]

Con la llegada de las nuevas tecnologías, se han abierto nuevas perspectivas en la resolución del problema planteado. Sobre todo teniendo en cuenta las tecnologías adaptadas para el uso casual o doméstico, tanto para mayores como para pequeños, como son los dispositivos móviles (teléfonos inteligentes, tabletas, etc.) que ahora más que nunca, están al alcance de todos.

Aun así, como se comprobará en el análisis del estado del arte siguiente, de los productos de software que comparten objetivos de la solución, todos estos artefactos o métodos con base en las nuevas tecnologías parten de los mismos principios e incentivos que los métodos tradicionales comentados anteriormente. Esto es debido a que si el producto de software pretende motivar a su jugador para realizar las tareas de limpieza del hogar, sigue necesitando un incentivo, entre otras cosas.

[9] Fuente: <http://es.aliexpress.com/item/Child-clean-toiletry-kit-baby-besmirchers-toy/1505939265.html/>.

2.2 Análisis del estado del arte de los productos de software que comparten los objetivos de la solución

En este apartado se analiza el estado del arte de los productos de software actuales más representativos, que compartan algún o algunos de los objetivos de la solución. Esto es, un análisis de las principales características de cualquier producto de software actual, no sólo videojuegos, que solucione o ayude a solucionar el problema. El objetivo de este análisis es determinar el género y las principales características base del videojuego a desarrollar, junto con un posible modelo de negocio para el mismo. Al finalizarlo se establecerán en el apartado de "Conclusiones para el análisis", justificando las decisiones tomadas en base al propio análisis.

Para llevar a cabo este análisis, inicialmente se realiza una búsqueda de aplicaciones, videojuegos y otros productos de software en motores de búsqueda, dividida en dos partes: **productos de software en general en PC** y **aplicaciones para dispositivos móviles**. El motivo de utilizar una búsqueda centrada en los dispositivos móviles, como se explicará más adelante, es que la mayoría de los jugadores del mercado casual, al que pertenece el videojuego a desarrollar, escogen estos dispositivos como sistema preferente [17]. Se intentará ser selectivo con el tipo de producto de software en los resultados escogidos para poder contrastar el máximo número de perspectivas de resolución del problema.

En la búsqueda de productos de software en general en PC, se utilizará el motor de búsqueda en internet más popular del momento: *Google* [26]. Para esta parte se introducen una serie de términos de búsqueda, combinaciones de palabras clave, con el fin de poder encontrar productos de software acordes al objetivo del análisis. Ejemplos de estos términos pueden ser: *"kids help house chores videogames"*, *"kid cleaning house software"*, *"videojuegos niños tareas domésticas"*, *"aplicación niños tareas de limpieza"* y similares. Igualmente se intenta refinar cada una de estas búsquedas utilizando variaciones de los términos (ej. Domésticas = del hogar, colaborar = ayudar, etc.). En cada búsqueda se escogen los productos más relevantes y representativos que se encuentren en las primeras páginas de resultados (máx. 3 páginas).

En el caso de la búsqueda de aplicaciones para dispositivos móviles en concreto, aparte de considerar los resultados derivados de la búsqueda anterior (ej. Aplicaciones de *App Store* de *Apple* aparecidas en los resultados de *Google*), se utiliza como motor de búsqueda la tienda del sistema operativo para dispositivos móviles más popular del momento: *Google Play Store* de *Android* [27]. Para realizar cada búsqueda se introducen términos similares que en la búsqueda anterior de productos de software en general en PC. En cada búsqueda se escogen los productos más relevantes y representativos que tengan más de 100000 descargas.

Figura 2.4: Rango de búsqueda para aplicaciones para dispositivos móviles

Una vez seleccionados los productos de software adecuados a partir de ambas búsquedas, se procede a analizar sus características principales de cara a la resolución del problema de forma individual.

2.2.1 Análisis de la web *Family Chores*

El primer producto de software a analizar es la web *Family Chores*, fundada por Patricia Gagnon y desarrollada por *Neo Vida Media Inc* [28].

Figura 2.5: Página principal de la web Family Chores con diagrama de flujo de su funcionamiento

A simple vista y como indica su sinopsis en la página principal, *Family Chores* es una web que permite a los padres asignar y organizar las tareas domésticas de sus hijos (no sólo tareas de limpieza). A los hijos por su parte les permite completar esas tareas, reflejándolo en la web y ganar puntos que podrán canjear por premios.

En la sección "*How it Works*" se muestra un diagrama de flujo más detallado de lo que ofrece la web:

Figura 2.6: Diagrama de flujo detallado del funcionamiento de Family Chores

Con esto en mente, se procede al registro de la familia para comprobar de primera mano, paso a paso, la experiencia que ofrece la web:

Primeramente, para registrar la familia, la web sólo permite registrar a uno de los padres, advirtiendo que el registro sólo puede ser realizado por un adulto. Éste es el que gestionará las tareas de sus hijos. Para ello hay que rellenar un formulario con los datos del padre o madre. Al terminar el registro, la web insta al usuario a obtener una cuenta en la página de compra *Amazon*, ya que los puntos que reciban los hijos al completar las tareas se traducirán en dinero real (1 punto = 0.10 $) que sólo podrá ser utilizado para comprar "premios" (artículos) en la web de compra *Amazon*.

A continuación, el padre o madre, ya inscrito e identificado en la página, puede acceder a un de panel de control que muestra información sobre la cuenta y le permite, entre otras cosas; crear, asignar y gestionar tareas domésticas y las cuentas de los hijos que tendrán que realizarlas. Para comprobar esta dinámica, se crea la cuenta de un hijo ficticio llamado "Pablitito". Para esto se tiene que rellenar otro formulario similar al de los padres, introduciendo los datos del hijo.

Figura 2.7: Panel de control paterno con un hijo añadido en Family Chores

Se procede a asignar una tarea a “Pablitito”. Ésta tarea puede ser escogida de una lista universal que contiene tareas de serie (ej. Limpiar el baño) junto con su valor en puntos al ser completada o puede ser creada desde cero (tarea personalizada). Igualmente se pueden establecer los días dentro de la semana en los que esa tarea debe ser realizada.

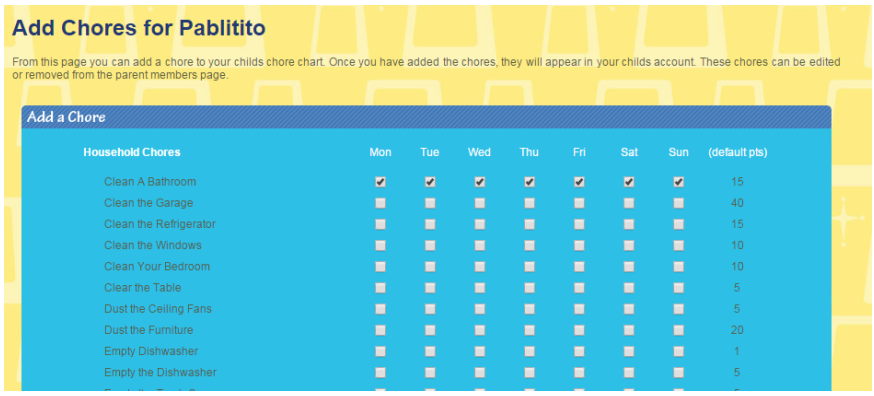


Figura 2.8: Asignación de tareas universales en Family Chores

Una vez que se ha creado la cuenta del hijo y se le ha asignado una o varias tareas, el padre o madre espera hasta que el hijo complete las tareas.

El hijo “Pablitito”, por su parte, se identifica en la página con la cuenta que creó el padre o madre, lo que le permite acceder a su propio panel de control, donde puede, entre otras cosas; ver las tareas (asignadas por el padre o la madre) que tiene pendiente de realizar, marcarlas como realizadas y establecer y elegir premios o recompensas que podrá canjear con los puntos que gana al completar las tareas.

Figura 2.9: Panel de control infantil junto con las tablas de tareas pendientes en Family Chores

Para establecer los premios que podrá canjear por puntos al completar tareas, los selecciona a través de un buscador de *Amazon* integrado. Esto significa que los premios que podrá obtener provienen exclusivamente de la tienda online *Amazon*. Al ser seleccionados se añaden a la "lista de deseos" del panel de control. Igualmente podrá canjear sus puntos por premios predeterminados, como dinero, tiempo para jugar a videojuegos, tiempo para ver la tele o incluso donarlos a la caridad.

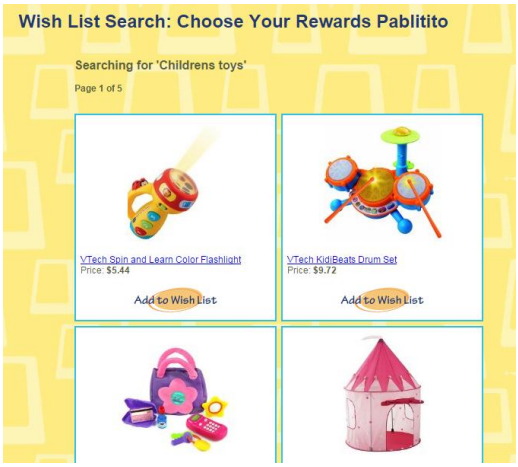


Figura 2.10: Buscador de premios en Family Chores

Teniendo en cuenta todo lo anterior, adivinar el resto del flujo es bien sencillo: Cuando "Pablitito" termina una de las tareas asignadas en la vida real, el padre o madre lo verifica desde su panel de control correspondiente, otorgando puntos en la cuenta de "Pablitito". El ciclo continúa hasta que "Pablitito" reúne los puntos suficientes para canjearlos por alguno de sus premios deseados.

Una vez comprobado el funcionamiento y el propósito de la web se pueden valorar sus principales características, junto con sus ventajas e inconvenientes.

Se puede observar que, a grosso modo, la web *Family Chores* no es más que una tabla de tareas digital levemente automatizada. "Levemente" ya que no presenta ningún método de reconocimiento automático, que podría evitar casos como el de que el padre o madre todavía tengan que comprobar físicamente si el hijo/a ha completado su tarea asignada, para confirmarla como completada en su panel de control. En otras palabras, todavía requiere introducir el progreso del flujo manualmente en la web. Su condición como tabla de tareas garantiza que si se usa correctamente, mediante su flujo, el hijo/a colaborará en las tareas del hogar.

Su principal atractivo se encuentra, como el resto de tablas de tareas tradicionales, en el incentivo, el premio que se promete al completar una tarea. En este caso incluso lo refuerza, ya que se permite al hijo/a elegirlo de una tienda (*Amazon*) y visualizarlo gráficamente como una meta a conseguir, como si hubiera ido físicamente a una tienda de juguetes real para elegirlo. La característica de los puntos, por su parte, hace que el hijo/a pueda visualizar con exactitud el progreso que va realizando y lo que le queda hasta llegar al premio, en contraste con las vagas promesas que podrían llegar a hacer los padres con una tabla tradicional. Todo esto aumenta de forma considerable su motivación de cara a realizar las tareas.

Igualmente, al tratarse de una tabla de tareas, comparte sus inconvenientes. Aparte de la promesa del incentivo, aunque en este caso poderoso, no presenta ninguna otra característica que haga las propias tareas más atractivas. Esto puede llevar a que el hijo/a vea las tareas como algo negativo y con el paso del tiempo puede provocar que se niegue a realizarlas, incluso si el premio es sustancial, ya que para él/ella no merece la pena tanto esfuerzo.

Tampoco ayuda, en este caso, el diseño del web, poco intuitivo, con demasiado texto y botones y pocas imágenes. Esto puede echar para atrás tanto a padres como a hijos, siendo especialmente tedioso a la hora de iniciarse con el funcionamiento y el registro en la web. Igualmente, la presencia constante de *Amazon* en características como la selección de premios, ya de por sí limitada, puede ser disuasoria para los padres. Esto es debido a que el servicio que ofrece la web es gratuito y su única fuente de ingresos proviene de un contrato que tienen con *Amazon*, que obliga a la web a ofrecer todos los premios o recompensas por la tienda online de *Amazon*.

2.2.2 Análisis de la plataforma *Choremonster*

El segundo producto de software a analizar es la plataforma web + aplicación móvil *Choremonster*, fundada y desarrollada por Chris Bergman and Paul Armstrong [29].

Figura 2.11: Página principal de la versión web de la plataforma Choremonster

La plataforma consta de una web y una aplicación *Android* para móvil, ambos gratuitos, que ofrecen exactamente el mismo servicio y mantienen las mismas funcionalidades, con capacidad de interactuar la una con la otra. En otras palabras se trata de dos versiones de un mismo servicio adaptadas para dos sistemas distintos.

El servicio que ofrece la plataforma, observando su sinopsis en la página principal, es muy similar al ofrecido por *Family Chores*. Al registrarse y probar su funcionamiento de primera mano se confirma que efectivamente su propósito, dinámica, funcionamiento y flujo es idéntico salvando ciertos detalles:

- Los puntos obtenidos por el hijo/a al completar una tarea no equivalen directamente a dinero a la hora de canjearlos por premios, simplemente se consideran como puntos.

- Los premios, por su parte, son creados por los padres mediante su panel de control, en vez de por los hijos como en *Family Chores*. En este caso los hijos escogen que premios desean canjear con sus puntos de una lista de premios creada por sus padres. Estos premios pueden ser creados desde cero, añadiendo su nombre, imagen y su valor en puntos o pueden ser escogidos de una lista de premios de serie. No limita la procedencia de los premios personalizados a *Amazon*.

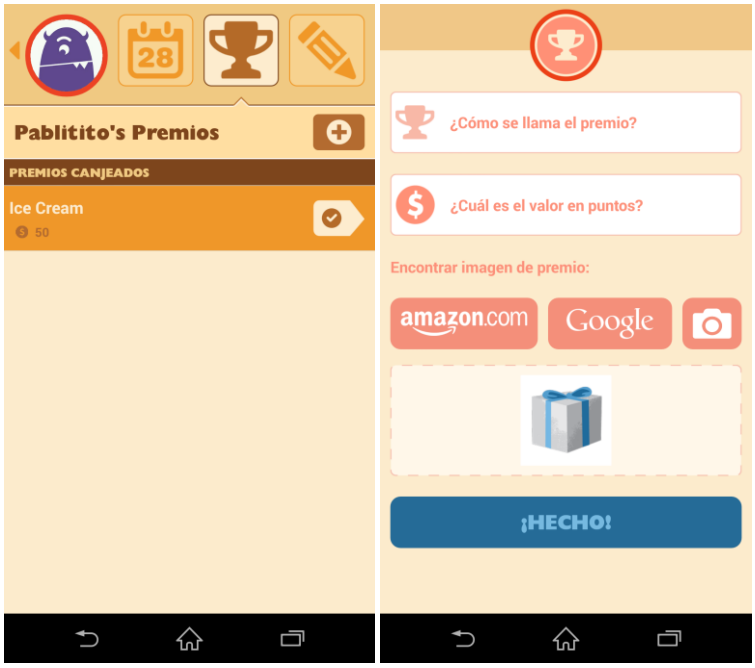


Figura 2.12: Creación de premios personalizados en la versión móvil de la plataforma Choremonster

- Aparte de los premios, cada vez que el hijo/a completa una tarea recibe un ticket para participar en el "Carnaval de los monstruos". Se trata de una sección o pantalla de la plataforma en la que el hijo/a puede probar suerte al girar una rueda de la fortuna. Cada ticket puede ser canjeado por un giro de ruleta, mediante el cual se puede ganar un monstruo coleccionable. Los monstruos que el hijo/a haya podido coleccionar se pueden visualizar en la sección de "Tus monstruos" junto con la biografía de cada uno de ellos.

Figura 2.13: Panel de control infantil y muestra del "Carnaval de monstruos" en Choremonster

- El diseño de la plataforma en general es minimalista, desenfado e intuitivo. Todos los elementos están construidos a base de figuras simples y vectoriales, utilizando texto sólo cuando es absolutamente necesario. Características como la tabla de tareas pendientes en el panel de control del hijo/a se ven simplificadas a botones vectoriales. El acercamiento inicial a la plataforma es mucho más intuitivo que en *Family Chores*; incluyendo un divertido y sencillo tutorial en su versión móvil que introduce tanto a los padres como al hijo/a al funcionamiento de la plataforma, registrándolos en el proceso.

Una vez comprobado el funcionamiento y el propósito de la plataforma se pueden valorar sus principales características, junto con sus ventajas e inconvenientes.

Debido a su similitud con *Family Chores*, *Choremonster* comparte la mayor parte de sus virtudes y defectos. Aunque en su página principal presumen que *Choremonster* no es otra tabla de tareas, posiblemente debido a su diseño intuitivo, realmente no deja de ser otra tabla de tareas digital levemente automatizada.

La principal característica que diferencia a *Family Chores* de *Choremonster* es como aborda el refuerzo del incentivo (todavía considerando el incentivo como el premio a conseguir). En el caso de *Family Chores*, como se comentó, permite a los hijos establecer y elegir de una tienda los premios que van a canjear por puntos, simulando el hecho de ir a una tienda físicamente para elegirlo y mantenerlo como meta a conseguir. *Choremonster* carece de esta característica, ya que son los padres los que establecen los premios, pero en su lugar cuenta con su característica insignia, el "Carnaval de monstruos". Mediante esta característica, se refuerza el incentivo del premio con otro incentivo extra: la posibilidad de obtener un monstruo para la

colección al girar la ruleta. Esto explota el afán de coleccionismo de los niños que ha conducido al éxito a otras franquicias como *Pokemon*[10].

Por otro lado, el diseño minimalista, intuitivo y desenfadado de la plataforma y su tutorial de uso, repleto de animaciones de los monstruos y otros aspectos infantiles, motiva al hijo/a a usar la plataforma. En contraste con los muros de texto y los paneles de control complejos de *Family Chores*. Igualmente, al contar con una versión para dispositivos móviles, cada vez más cercanos a los niños, refuerza su motivación para el uso.

Como principal inconveniente mantiene el mismo que *Family Chores*, inherente a las tablas de tareas: No hace atractivas las propias tareas para el usuario. Confía en gran medida en el incentivo, el premio más el carnaval de monstruos, como excusa para que el usuario realice la tarea. Esto puede no ser suficiente para algunos niños, que seguirán considerando estas tareas como pura obligación o algo negativo.

2.2.3 Análisis de las aplicaciones Abby's Home Laundry y Baby Home Adventure

El tercer análisis se realizará sobre las aplicaciones gratuitas *Android Abby's Home Laundry*, desarrollada por O*zone Development* [30] y *Baby Home Adventure,* desarrollada por *Tabtale* [31]. Se decide agrupar ambos en un análisis debido a que presentan unas características muy similares.

Abby's Home Laundry es el título de un videojuego educativo para *Android* que simula todo el proceso de hacer la colada. Al iniciar el juego, mediante una introducción en forma de texto, se informa al jugador que tiene la misión de ayudar a Abby, protagonista del juego, a hacer la colada para después poder ponerse ropa limpia e ir a una fiesta.

Dentro del juego la interacción del jugador, o jugabilidad, se reduce a mecanismos *drag and drop*[11] y ciertas pulsaciones de elementos a través de la pantalla táctil. Mediante estos mecanismos se emulan acciones como llevar una prenda sucia hasta su cesta correspondiente, verter el detergente adecuado en la lavadora o encender la misma. Estos controles son intuitivos ya que simulan de forma análoga la reacción instintiva que tendría un jugador al encontrarse en una situación similar (ej. Recoger un juguete de un sitio y llevarlo hasta otro, se traduce en pulsar ese juguete en la pantalla táctil y arrastrarlo, manteniendo esa pulsación, hasta el punto deseado, donde se suelta la pulsación para soltar el juguete).

El juego se desarrolla en 5 fases que emulan todo el proceso de hacer la colada. En la primera fase, el jugador tiene que organizar la habitación de Abby, identificando las prendas de ropa sucia blanca y de color y depositándolas en las cestas correspondientes. En la segunda, el jugador tiene introducir las prendas sucias de

[10] Fuente: <http://www.pokemon.com/es/>.
[11] Mecanismo que permite arrastrar elementos virtuales por la pantalla, agarrándolos para luego soltarlos.

cada cesta en la lavadora y usar el detergente adecuado para cada tipo. En la tercera el jugador tiene tender la ropa en el tendedero del jardín, cada prenda en el lugar indicado. En la cuarta el jugador debe doblar toda la ropa y meterla en el armario de la habitación de Abby. En la quinta y última fase, se le permite al jugador vestir a Abby para la fiesta, con las prendas que ha limpiado durante el proceso.

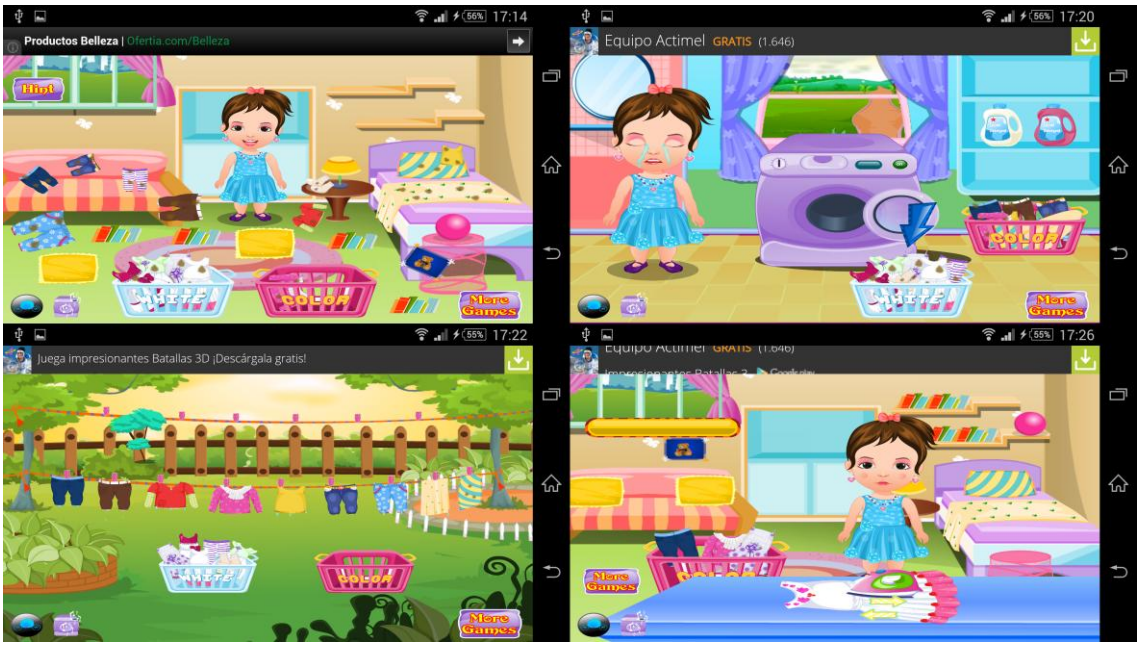


Figura 2.14: Proceso de hacer la colada en Abby's Home Laundry

Baby Home Adventure, por su parte, trata de una compilación de mini-juegos que representan tareas domésticas del hogar. Como la mayoría de los mini-juegos están inicialmente bloqueados, con la posibilidad de comprarlos aparte con dinero real, sólo se analizará el único mini-juego gratuito sobre la limpieza en el hogar: hacer la colada. De todas formas este modelo de negocio modular con micro pagos se tendrá en cuenta a la hora de establecer un posible modelo de negocio para el videojuego a desarrollar.

El mini-juego sigue la misma premisa que *Abby's Home Laundry*, con una jugabilidad idéntica, combinando *drag and drop* con pulsaciones y presentando una temática similar en general. Las únicas diferencia notables, es que en vez de involucrar al jugador con un trasfondo en tercera persona (ej. "Ayudar a Abby") el juego propone al jugador seleccionar un bebé como avatar en el juego. En otras palabras, el jugador realiza las tareas en la piel del bebé. Por otro lado, el mini-juego es más corto que *Abby's Home Laundry*, presentando sólo las 2 primeras fases; identificar, recoger y separar en cestas la ropa de color de la blanca y meterlas en la lavadora con el detergente correspondiente.

Figura 2.15: Flujo del minijuego de hacer la colada en Baby Home Adventure

Ambos videojuegos presentan una perspectiva en primera persona y unos gráficos 2d simples y coloridos, con animaciones suaves y fluidas estilo flash. Todos los personajes y elementos están diseñados de forma exagerada y divertida, como si fueran caricaturas o dibujos animados. La música y los efectos de sonido siguen el mismo estilo caricaturesco y divertido.

Una vez comprobado el funcionamiento y el propósito de las aplicaciones se pueden valorar sus principales características, junto con sus ventajas e inconvenientes.

Es sencillo ver que tanto *Abby's Home Laundry* como *Baby Home Adventure* son videojuegos educativos cuyo objetivo es fomentar la realización de las tareas del hogar, más concretamente de hacer la colada. Sus sencillos e intuitivos controles, su estética y su trasfondo y la ausencia de penalizaciones (los niños no pueden perder en el juego) están diseñados para atraer a jugadores infantiles y hacer que vean este tipo tareas de forma positiva, como un juego o desafío. El hecho de dar un trasfondo al juego, ya sea con la misión de ayudar a Abby o ayudar en casa como un avatar de bebé, aporta un sentido de la responsabilidad o involucración al jugador, que pasa a ver estas tareas como algo necesario de hacer y positivo, incluso sin esperar ningún premio a cambio.

El principal inconveniente que tienen ambos videojuegos es que, a diferencia de los productos de software tipo tablas de tareas anteriormente comentados, no garantizan que el jugador realice verdaderamente las tareas de limpieza en la vida real. Estos juegos carecen del incentivo del premio, y sólo fomentan la realización de las tareas, haciéndolas ver como algo positivo, un juego o desafío. En otras palabras, independientemente de las ventajas e inconvenientes o el éxito que tengan, mediante el uso de los productos de software tipo tablas de tareas se soluciona directamente el problema, mientras que con este tipo de videojuegos sólo se fomenta la solución.

Por otra parte, la jugabilidad sencilla y la poca profundidad de ambos videojuegos es un arma de doble filo, ya que de la misma forma que atrae a sus jugadores objetivos (niños) de forma inicial, puede hacer que pierdan el interés a largo plazo, debido a que el videojuego en general se hace repetitivo.

Como detalle final, el hecho de que todos los personajes de ambos juegos, jugables y no jugables, sean del género femenino puede traer connotaciones sociales negativas. Esto se acentúa en *Baby Home Adventure*, donde se permite al usuario elegir un avatar en forma de bebé, ya que debería contener algún avatar de género masculino para garantizar una representación igualitaria.

2.2.4 Análisis de las aplicaciones Kids House Clean Up y Toca House

El cuarto análisis se realizará sobre la aplicación gratuita *Android Kids House Clean Up*, desarrollada por *Mobile Games Media* [32] y sobre la aplicación de pago *Toca House,* desarrollada por *Toca Boca AB* [33]. Se decide agrupar ambos en un análisis debido a que presentan unas características muy similares.

Se trata de dos videojuegos educativos con una premisa muy similar a los comentados en el apartado anterior; pero con la diferencia de que en *Kids House Clean Up* el jugador debe ayudar a Alaina (personaje no jugable) a limpiar su casa, mientras que en *Toca House* el jugador debe ayudar a los 5 habitantes de una casa en una compilación de mini-juegos relacionados con las tareas domésticas (no sólo de limpieza).

De igual forma, la jugabilidad de ambos juegos es muy similar a los anteriormente comentados, haciendo uso de mecánicas estilo *drag and drop* y pulsaciones de formas creativas de acuerdo a la acción necesaria a realizar. Por ejemplo, a la hora de utilizar un trapo para limpiar manchas en la pared, se coge el trapo pulsando sobre él y se restriega por la mancha, moviendo el trapo de un lado para otro mediante *drag and drop*, hasta que la mancha gradualmente desaparece.

En *Kids House Clean Up*, el juego comienza con una breve introducción de texto, explicando el motivo de tener que ayudar a Alaina de forma educativa, haciendo alusión a las necesidades de higiene que todos los niños deberían tener. El juego se desarrolla a través de 3 fases, que representan diversas partes de la casa de Alaina que hay que limpiar: su cuarto, la cocina y el jardín. En cada fase, el jugador deberá ayudar a Alaina ordenando los objetos que haya desordenados, limpiando dibujos y manchas que pueda haber en las paredes y el suelo con un trapo o una aspiradora y tirando a la papelera papeles arrugados que ensucien el suelo. Inicialmente todas las fases, excepto la primera están bloqueadas, teniendo que completar la fase anterior a cada una para desbloquearlas.

La principal diferencia de *Kids House Clean Up*, respecto a *Toca House* y a los comentados en el apartado anterior, es el añadido de una barra de tiempo. Si el jugador no ordena la habitación a tiempo, se le penaliza teniendo que volver a empezar, con todo como estaba al principio.

Estéticamente *Kids House Clean Up* es idéntico a *Abby's Home Laundry* y *Baby Home Adventure*, con gráficos 2d caricaturescos y coloridos, estilo flash, con animaciones fluidas propias del mismo estilo.

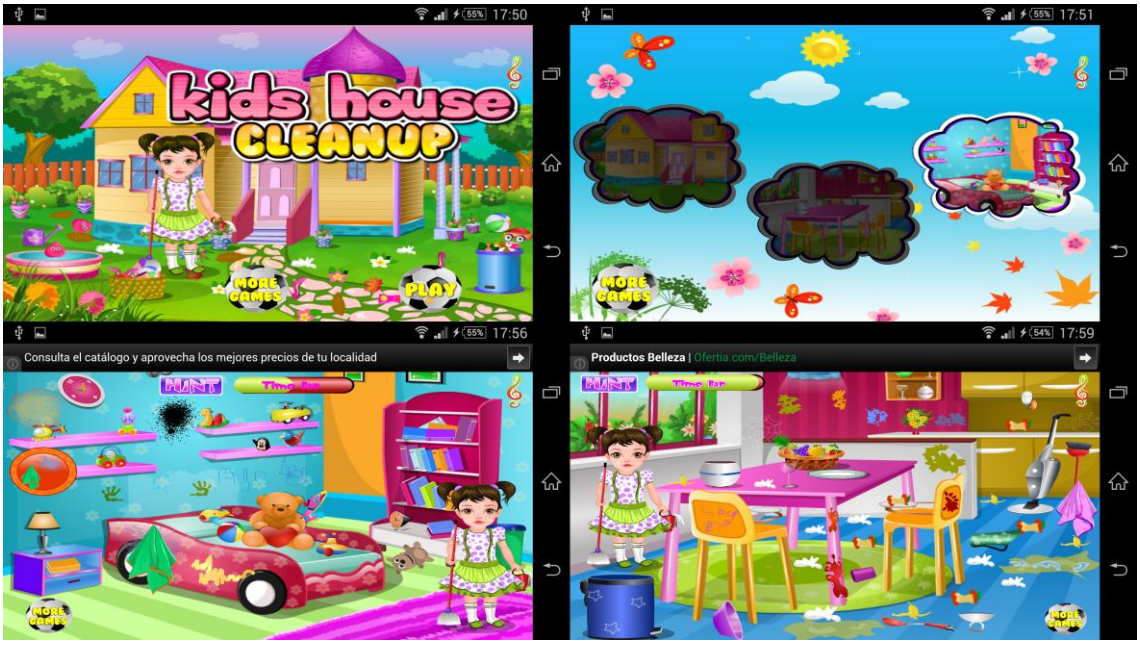


Figura 2.16: Flujo de las fases en Kids House Clean Up y muestra de jugabilidad

Toca House, por su parte, comienza con una animación como si de una serie de dibujos animados se tratase, poniendo al jugador al corriente de lo que debe realizar. El juego se desarrolla en una casa, donde el jugador puede ver lo que sucede en cada una de sus cinco habitaciones. Cada habitación representa una fase, donde se encuentra un personaje no jugable que el jugador deberá ayudar a completar alguna tarea del hogar mediante un mini-juego que varía según la habitación.

Al contrario que *Kids House Clean Up, Toca House* no penaliza de ninguna forma al jugador, no hay ningún tipo de retroalimentación negativa.

La principal característica de *Toca House*, que le diferencia del resto de juegos con una premisa similar, es su diseño visual y extremadamente intuitivo. Los elementos y personajes del juego están caracterizados con un estilo 2d único y original (formas vectoriales, colores pastel, trazos bruscos, etc.). El juego en general carece de texto y de menús, conduciendo a sus jugadores a base de fluidas animaciones por todas las pantallas del juego. No hay tutoriales, ni indicaciones de lo que el jugador deber realizar en cada momento, el juego es lo suficientemente intuitivo como para que el jugador se dé cuenta mediante el contexto visual. De igual forma el trasfondo de los personajes, no se explica directamente, dejándolo a la imaginación del jugador en base al contexto visual. Todo esto le aporta una personalidad única, muy atractiva a los ojos de los niños.

Figura 2.17: Muestra de la jugabilidad e imágenes promocionales de Toca House

Una vez comprobado el funcionamiento y el propósito de las aplicaciones se pueden valorar sus principales características, junto con sus ventajas e inconvenientes.

Tanto *Kids House Clean Up* como *Toca House* presentan una premisa y jugabilidad muy similar a los juegos comentados anteriormente. Son juegos educativos que, en este caso, fomentan la realización de las tareas de limpieza y del hogar en general. Por lo tanto mantienen el mismo dilema clave de este tipo de juegos: Mientras que mediante su uso fomentan de forma positiva la realización de las tareas de limpieza en el hogar en la vida real, no garantizan que efectivamente se realicen como con los productos de software tipo tablas de tareas.

Obviando las ventajas e inconvenientes inherentes de este tipo de juegos, ya comentadas en el apartado anterior, como aspectos destacables cabe mencionar la forma que tiene cada juego de abordar positivamente las tareas del hogar. En el caso de *Kids House Clean Up*, la inclusión de una barra de tiempo que penaliza al jugador, hace que el juego sea más desafiante y adictivo, aportando profundidad a la jugabilidad. Esto por un lado ayuda a que los jugadores puedan considerar las tareas en la vida real como un desafío o reto positivo, pero por otro puede espantar a jugadores de menor habilidad, debido al aumento de dificultad. En el caso de *Toca House,* como se comentó anteriormente, todo el conjunto de gráficos, animaciones, caracterización de los personajes y ambientación en general, hacen que el videojuego tenga un trasfondo original y una personalidad propia. Este aspecto hace que el juego sea altamente atractivo a los ojos de los niños y demuestra cómo, con un buen trasfondo y ambientación, se puede involucrar aun más a los jugadores con las situaciones del juego y hacer llegar el mensaje que quiere mandar: “Las tareas del hogar son divertidas y realizándolas se puede hacer feliz a mucha gente”.

2.2.5 Conclusiones del análisis

Una vez realizado el análisis del estado del arte los productos de software que comparten los objetivos de la solución, se procede a formular las conclusiones del mismo. El objetivo de ello es establecer el género y las principales características base del juego a desarrollar, junto con un posible modelo de negocio para el mismo.

Con las características base del videojuego establecidas, se puede determinar finalmente el género del videojuego. Para ello se presentan estas conclusiones divididas por puntos, cada uno de ellos representando los aspectos o características base de un videojuego convencional: premisa, dirección artística (gráficos y estilo más sonido), jugabilidad, trasfondo y por último el propio género.

Premisa

A través del análisis es sencillo identificar dos grupos o tipos bien diferenciados en los que se pueden clasificar los productos de software analizados, de acuerdo a la forma en que abordan la solución del problema planteado: productos de software tipo tablas de tareas y productos de software tipo simulación. *Family Chores* y *Choremonster* pertenecerían al primer tipo, mientras que el resto pertenecerían al segundo.

La diferencia entre los dos tipos es que, mediante el uso de los productos de software tipo tabla de tareas, se garantiza la colaboración en las tareas de limpieza en el hogar, pero no se fomenta de forma positiva la propia realización de las mismas. En otras palabras, este tipo de productos apela a la motivación extrínseca de los niños para realizar las tareas, valiéndose únicamente del premio que recibirán al completar la "tabla de tareas" como incentivo para realizarlas. Por el contrario, mediante el uso de los productos de software tipo simulación se fomenta de forma positiva la realización de estas tareas, pero no se garantiza la colaboración en las mismas. En otras palabras, este tipo de productos apelan a la motivación intrínseca de los niños, presentando las tareas que "simulan" como algo divertido o algo de carácter responsable, necesario de hacerse sin buscar nada a cambio y con el fin de hacer feliz a los personajes del juego (y de forma análoga en la vida real, a sus padres o tutores).

Teniendo en cuenta esto, para que el juego a desarrollar pueda ser utilizado para solucionar el problema planteado de la forma más óptima, necesita presentar los aspectos beneficiosos de ambos tipos. Esto es, necesita fomentar de forma positiva las tareas de limpieza del hogar y garantizar su realización.

Para ello, el videojuego a desarrollar debe, por un lado, simular la realización de tareas de forma positiva, como los productos de software de tipo simulación. En este caso, debe simular la realización de varias tareas de limpieza del hogar, cada una con diferentes características. Una forma optima de implementar en el videojuego a desarrollar este tipo de simulación, atendiendo a la organización de las tareas en *Baby Home Adventure* y *Toca House*, es mediante la distribución de las tareas de limpieza en mini-juegos individuales que las representen.

Por otro lado, el videojuego debe garantizar que las tareas de limpieza en el hogar efectivamente sean realizadas por los jugadores en la vida real. Como se ha comentado en apartados anteriores, esto solo es posible mediante un incentivo hacia el jugador, prometiendo algún tipo de premio al completar una tarea en la vida real. Tanto *Family Chores* como *Choremonster* representan excelentes ejemplos de cómo llevar acabo esto. *Choremonster* en particular, presenta la exitosa característica del "Carnaval de monstruos", que ofrece la posibilidad de obtener monstruos coleccionables dentro de la aplicación al completar una tarea en la vida real. En el videojuego a desarrollar se podría implementar un sistema similar, prometiendo al jugador algún tipo de elemento coleccionable virtual al completar una tarea de limpieza en la vida real. El hecho de que el elemento coleccionable sea virtual en vez de un premio real, permite que pueda ser integrado con la faceta de simulación del juego anteriormente comentada, pudiendo obtener estos elementos igualmente al completar los mini-juegos que conforman el juego, aumentando su atractivo.

Al considerar las dos facetas del videojuego a desarrollar juntas, su premisa pasa a ser la siguiente: Simulación educativa de las tareas de limpieza del hogar mediante mini-juegos, los cuales al ser completados otorgan elementos virtuales coleccionables que igualmente podrán ser conseguidos realizando tareas de limpieza en el hogar en la vida real mediante un sistema de realidad aumentada. "Realidad aumentada", ya que el juego deberá interactuar de forma directa o indirecta con el entorno físico del mundo real del jugador, para reconocer si las tareas efectivamente han sido realizadas en la vida real.

Dirección artística

Todos los productos de software analizados, en especial los videojuegos, presentan en mayor o menor medida gráficos 2d simples y coloridos, con animaciones fluidas y estilo desenfadado de dibujo animado infantil en general. El diseño de todos sus elementos es simple e intuitivo, ya que apoya a la jugabilidad (mediante el diseño de los elementos se dan indicaciones sutiles al jugador de cómo jugar el videojuego). Igualmente todas sus músicas de fondo son similares a melodías o canciones de estilo infantil con sonidos divertidos que acompañan cada situación.

Toda la dirección artística en estos productos está diseñada para ser más atractiva a su público infantil objetivo y tener más posibilidades de ser utilizados.

El videojuego a desarrollar deberá presentar una dirección artística con unas características similares pero tomando ciertas libertades para encontrar un estilo único que le aporte personalidad al juego, aumentando su atractivo, como ya hizo *Toca House*. Por ejemplo se considerarán otras alternativas al aspecto 2d de los gráficos, ya que posiblemente hayan sido fruto de limitaciones del motor gráfico o de la inexperiencia de su desarrollador, en pos de buscar esa personalidad única.

Jugabilidad

Este aspecto solo se aplica a los productos de software de tipo simulación o videojuegos. Todos los videojuegos analizados, a causa de las características de su público infantil objetivo, presentan una jugabilidad y dinámica simple y

extremadamente intuitiva, apoyada por la dirección artística. El esquema de control es prácticamente común en todos ellos, haciendo uso de una combinación de mecánicas *drag and drop* y pulsaciones en la pantalla táctil de forma creativa, adaptada a cada situación que se presente en los juegos.

El problema es que una jugabilidad demasiado simple puede llegar a aburrir a los usuarios objetivos, dejando de fomentar de forma positiva la tarea. La adición de elementos de penalización como la barra de tiempo en *Kids House Clean Up* pueden dar profundidad a esta jugabilidad si son usados sabiamente.

La jugabilidad del juego a desarrollar deberá ser similar a la de todos estos juegos, utilizando mecanismos simples adaptados para cada tipo de tarea de limpieza. Presentará elementos similares a la barra de tiempo, que otorguen profundidad a la jugabilidad, pero controlando la penalización en pos de que los usuarios con menos habilidad no se sientan rechazados. De igual manera se tomarán ciertas libertades con algunos aspectos de la jugabilidad, con el objetivo de encontrar una personalidad única para el juego, acompañando la dirección artística y aumentando su atractivo de cara a los usuarios objetivos.

Trasfondo

El trasfondo[12] de un videojuego, junto con la dirección artística, es esencial a la hora de aumentar el atractivo del producto hacia el público objetivo. Todos los productos analizados, excepto la web *Family Chores*, poseen en mayor o menor medida un trasfondo que los completa, pero es en los productos de tipo simulación o videojuego donde se hace más patente el efecto que provoca.

Todos los trasfondos de los productos de tipo simulación analizados tienen una estructura similar: El jugador, como él mismo o como otro personaje o avatar dentro del juego, por alguna razón justificada, debe realizar tareas del hogar por sí mismo o ayudando a otro personaje. La acción se desarrolla dentro de una casa u otro entorno doméstico familiar y atractivo para el jugador. Este trasfondo está diseñado para involucrar de forma inconsciente al jugador objetivo, aumentando su sentido de la responsabilidad y fomentando estas tareas de forma positiva al usuario, que pasará a verlas en la vida real como un juego o desafío o algo que es necesario hacerse incluso sin esperar nada a cambio.

De igual forma, la manera de presentar este trasfondo a los jugadores debe ser la adecuada. Mientras que en juegos como *Abby's Home Laundry*, apuestan por explicar gran parte del trasfondo mediante una introducción de texto inicial, en juegos como *Toca House* el trasfondo se va mostrando de forma sutil durante el desarrollo del juego, utilizando la dirección artística. La forma adecuada debe ser elegida de acuerdo a las características del jugador objetivo, como por ejemplo el rango de edad y su capacidad lectora.

[12] En el contexto de los videojuegos, se entiende como trasfondo de un videojuego como cualquier característica "decorativa" del mundo o entorno simulado del videojuego, que no aporte información sobre cómo jugarlo (instrucciones directas o indirectas).

El trasfondo del videojuego a desarrollar deberá seguir la misma estructura comentada, sin abandonar algunos aspectos originales que puedan inferir personalidad propia al videojuego y hacerlo más atractivo a los ojos de los usuarios objetivos. La forma de presentar este trasfondo, atendiendo a las características del usuario objetivo es una combinación acertada de los métodos mencionados, esto es, mediante la dirección artística del juego (escenarios, caracterización de los personajes, etc.) acompañada de texto o voz superpuesta cuando sea necesario.

Género

Una vez establecidas todas las características base, es posible determinar el género del videojuego a desarrollar. El género del videojuego a desarrollar es el de **compilación de mini-juegos educativos con funciones de realidad aumentada.**

2.2.6 Resultados del análisis desglosados

A continuación se muestra una tabla con las características base resumidas del videojuego a desarrollar fruto del análisis del estado del arte los productos de software que comparten los objetivos de la solución:

Característica base	Descripción resumida	Producto software de origen
Premisa	*Simulación educativa de las tareas de limpieza del hogar mediante mini-juegos, los cuales al ser completados otorgan elementos virtuales coleccionables que igualmente podrán ser conseguidos realizando tareas de limpieza en el hogar en la vida real mediante un sistema de realidad aumentada.*	Todos los analizados.
Dirección artística	*Gráficos simples y coloridos, con animaciones fluidas y estilo desenfadado de dibujo animado infantil en general. El diseño de todos sus elementos es simple e intuitivo ya que apoya a la jugabilidad (mediante el diseño de los elementos se dan indicaciones sutiles al jugador de cómo jugar el videojuego). Igualmente todas sus músicas de fondo son similares a melodías de estilo infantil con sonidos divertidos que acompañan cada situación.*	*Choremonster, Abby's Home Laundry, Baby Home Adventure, Kids House Clean Up, Toca House.*
Jugabilidad	*Jugabilidad y dinámica simple y extremadamente intuitiva, apoyada por la dirección artística. El esquema de control se basa en una combinación de mecánicas drag and drop y pulsaciones en la pantalla táctil de forma creativa, adaptada a cada situación que se presente en los juegos. Presentará elementos desafiantes similares que otorguen profundidad a la jugabilidad, como una barra de tiempo o un sistema de puntos, pero controlando la penalización en pos de que los usuarios con menos habilidad no se sientan rechazados.*	*Abby's Home Laundry, Baby Home Adventure, Kids House Clean Up, Toca House.*
Trasfondo	*El jugador, como él mismo o como otro personaje o avatar dentro del juego, por alguna razón justificada, debe realizar tareas de limpieza por sí mismo o ayudando a otro personaje. La acción se desarrolla dentro de una casa u otro entorno doméstico familiar y atractivo para el jugador. La forma de presentar este trasfondo es mediante la dirección artística del juego (escenarios, caracterización de los personajes, etc.) acompañada de texto o voz superpuesta cuando sea estrictamente necesario.*	*Choremonster, Abby's Home Laundry, Baby Home Adventure, Kids House Clean Up, Toca House.*

Tabla 2.1: Tabla de las características base del videojuego a desarrollar

A continuación se muestra una tabla centrada en la influencia aproximada que han tenido los productos de software analizados sobre cada característica base del videojuego a desarrollar:

Productos de software analizados	Premisa	Dirección artística	Jugabilidad	Trasfondo
Family Chores	●	-	-	-
Choremonster	●	●	-	●
Abby's Home Laundry	●	●	●	●
Baby Home Adventure	●	●	●	●
Kids House Clean Up	●	●	●	●
Toca House	●	●	●	●

Leyenda: ● Especialmente influenciado; ● Muy influenciado; ● Poco influenciado; – No influenciado

Tabla 2.2: Tabla de influencias del videojuego a desarrollar

2.2.7 Modelo de negocio base

El modelo de negocio definitivo para el videojuego a desarrollar, no se puede establecer hasta la fase de postproducción (apartado 3.8), cuando se conozcan las características finales del prototipo de software de alto nivel de fidelidad. Aun así, mediante el análisis del estado del arte los productos de software que comparten los objetivos de la solución es posible esbozar un modelo de negocio base, del cual se partirá en la fase de postproducción.

Los modelos de negocio de los productos de software a considerar para esbozar el modelo de negocio base son los de los productos de tipo simulación o videojuegos, ya que el videojuego a desarrollar entra dentro de esta categoría. En este caso, los analizados presentan 3 tipos de modelos de negocio bien diferenciados:

- Distribución gratuita, con publicidad como fuente de ingresos.
- Distribución gratuita, con la inclusión de micro pagos opcionales futuros.
- Distribución de pago.

Se ignora de momento el tipo de plataforma de distribución, ya que será decidido en la postproducción. En su lugar, se considera una plataforma genérica.

Teniendo en cuenta el género y las características base del juego establecidas, el modelo de negocio de **distribución gratuita (sin publicidad) con la inclusión de micro pagos o micro transacciones opcionales futuras** es la mejor opción. La justificación de esta decisión puede separarse en 4 puntos:

- La premisa del juego a desarrollar, compilación de mini-juegos educativos, junto con su característica de ofrecer elementos coleccionables como premios, le otorgan una condición modular, perfecta para este tipo de modelo de negocio. Mientras que el videojuego base, con un número de mini-juegos y elementos coleccionables por defecto, se distribuye de forma gratuita, el jugador tiene la posibilidad de comprar con dinero real mini-juegos y elementos coleccionables extras que complementen el videojuego base.

- Una distribución gratuita es mucho más atractiva al jugador casual que una distribución de pago. Los padres o tutores de los niños no deberían tener inconveniente en adquirir el juego base, obteniendo así más clientes potenciales para los micro pagos y reforzando el propósito del videojuego (solucionar el problema planteado).

- Este tipo de modelo de negocio ha sido el responsable de llevar videojuegos como los fenómenos mediáticos *Candy Crush Saga*[13] o *League of Legends*[14] al éxito y es visto por muchos como el modelo de negocio del futuro para los videojuegos en general [34].

- La "*Ley 34/1988, de 11 de noviembre, General de Publicidad*" [35], expone que la publicidad dirigida a un público infantil (menores de edad) puede llegar a tener una gran influencia sobre ellos debido a su inexperiencia o ingenuidad. Por ello, este tipo de publicidad debe estar estrictamente regulada o podría llegar a tacharse como ilegal. Por este motivo se ha decidido que, para evitar problemas legales debido al rango de edad de los usuarios objetivos, es más adecuado evitar utilizar publicidad integrada en el videojuego a desarrollar y centrar la fuente de beneficios únicamente en los micro pagos.

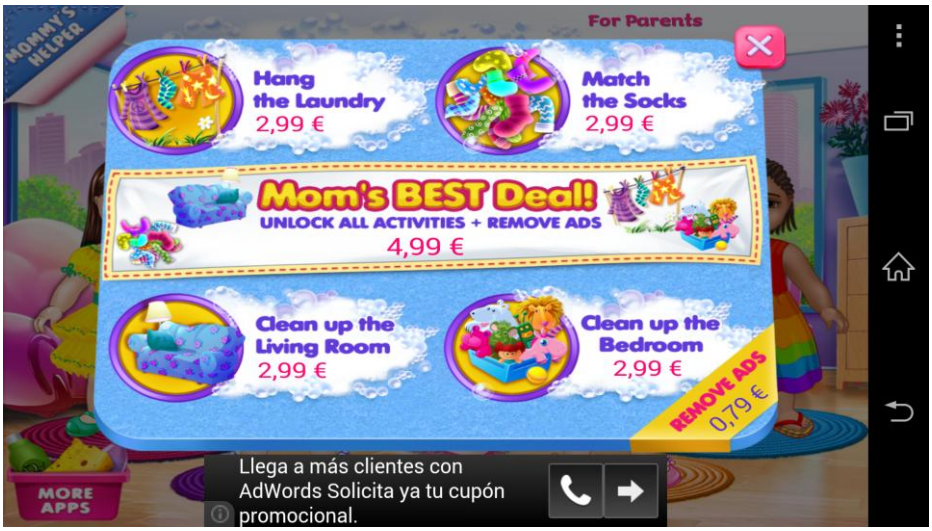


Figura 2.18: Tienda de micro pagos integrada en el videojuego gratuito Baby Home Adventure

[13] Fuente: <http://en.wikipedia.org/wiki/Candy_Crush_Saga>.
[14] Fuente: <http://en.wikipedia.org/wiki/League_of_Legends>.

2.3 Análisis del estado del arte del sistema y del kit de desarrollo de videojuegos

En este apartado se analiza el estado del arte de los sistemas actuales más relevantes utilizados para soportar videojuegos con las mismas características base que el videojuego a desarrollar. El objetivo es determinar el sistema adecuado para albergar el mismo. Una vez determinado, se realizará otro análisis del estado del arte de los kits de desarrollo de videojuegos que se utilizan actualmente en ese sistema para poder determinar el más adecuado para el desarrollo.

2.3.1 Análisis del sistema

Actualmente se utilizan diversos sistemas como soporte para videojuegos, tanto hardware como software. Entre ellos se cuentan sistemas como consolas de sobremesa, consolas portátiles, ordenadores de sobremesa o dispositivos móviles con diferentes sistemas operativos, etc. En este análisis se podrían considerar todos ellos como candidatos para el videojuego a desarrollar, pero debido a la condición del mismo como videojuego tipo casual (y dentro de casual, educativo), es posible limitar ésta búsqueda.

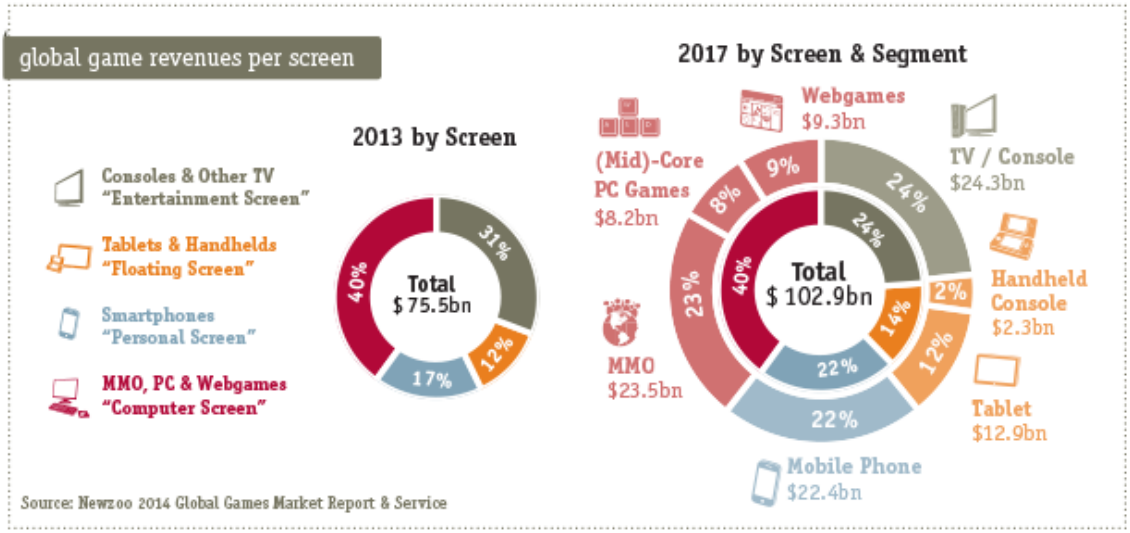


Figura 2.19: Estimación de beneficios globales de la industria del videojuego del 2013 a 2017[15]

Respecto al sistema hardware, como ya se apuntó en el apartado de "Motivación" (apartado 1.3), en la actualidad y de cara al futuro los **dispositivos móviles**, como los ***Smartphone* o *Tablets***, son el sistema hardware predominante y con más potencial en el mercado de los juegos tipo casual. Para confirmarlo sólo es necesario comparar los datos sobre el tipo de videojuego más jugado en dispositivos móviles desde que

[15] Fuente: <https://s3.amazonaws.com/CGA_Report/CCNewzooSpringReport-pages.pdf>.

comenzó el boom de los videojuegos tipo casual, iniciado por *Nintendo* en el 2012 [16], con el porcentaje de mercado que ocupan los propios dispositivos móviles.

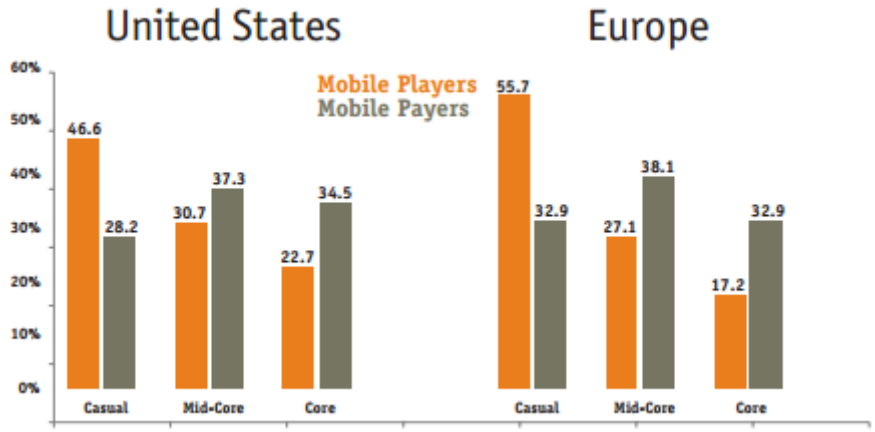


Figura 2.20: Desglose de tipos de usuarios de móviles en EEUU y Europa (2013)[16]

Figura 2.21: Tipos de juegos para móviles más jugados en EEUU (2012 – 2013)[17]

Aunque dentro de la clasificación de "dispositivos móviles" entrarían sistemas embebidos como las consolas de videojuegos portátiles (ej. *Nintendo DS*), se descartan debido a la poca presencia que tienen dentro de esta clasificación, considerando sólo dispositivos móviles como *Smartphone* y *Tablets.* De igual forma, los niños del rango de edad propuesto para el videojuego a desarrollar (de 6 a 9 años) suelen preferir dispositivos móviles, como *Tablets*, a la hora de jugar a este tipo de videojuegos.

[16] Fuente:
<https://s3.amazonaws.com/CGA_Report/CGA_Market_Report_Fall2013.pdf>.

[17] Fuentes:
<http://www.theesa.com/wp-content/uploads/2014/10/ESA_EF_2014.pdf> y
<https://www.theesa.com/facts/pdfs/ESA_EF_2013.pdf>.

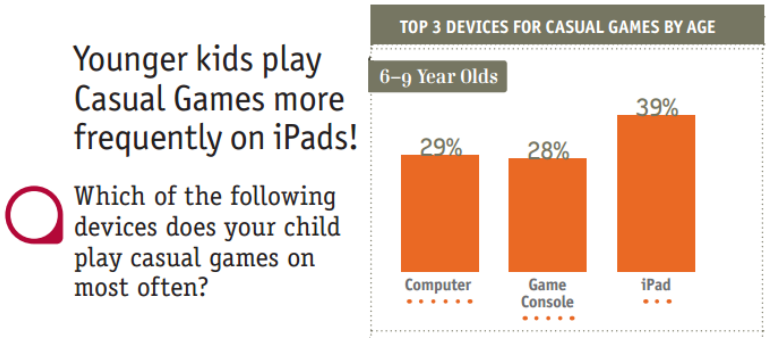


Figura 2.22: Top 3 de los dispositivos en los que los niños juegan a videojuegos casuales (2012)[18]

Por lo tanto, es factible considerar los dispositivos móviles como el sistema hardware que albergue el videojuego a desarrollar.

Una vez escogido el dispositivo hardware, es necesario determinar un sistema operativo software que lo complemente. Actualmente, en el mercado de los dispositivos móviles se encuentran múltiples sistemas operativos a considerar: *Android*, *iOS*, *Java ME*, *Symbian*, *Windows Phone*, etc. Al comprobar los últimos datos sobre el porcentaje de mercado que ocupan cada uno de ellos se puede observar que tanto *Android* como *iOS* son los más populares o utilizados.

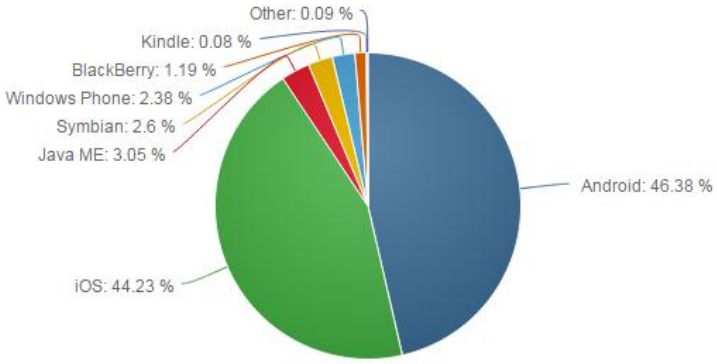


Figura 2.23: Presencia global de los SO para dispositivos móviles en el mercado (Octubre 2014)[19]

Aun estando más o menos igualados se decide utilizar el **sistema operativo *Android*** como sistema operativo objetivo por los siguientes motivos:

- ***Android* es el sistema operativo dominante en el mercado de videojuegos tipo casual**. Ya sea por la libertad y comodidad que *Google* deja a los desarrolladores para publicar sus productos en su tienda oficial, *Google Play Store*, o por la accesibilidad que tienen los usuarios de distintas marcas de dispositivos móviles

[18] Fuente: <*https://dl.dropboxusercontent.com/u/3698805/Reports/kidsreport2013.pdf*>.

[19] Fuente: <http://www.netmarketshare.com/>.

hacia este sistema operativo, datos recientes demuestran que *Android* es el sistema operativo donde se juegan más videojuegos de tipo casual.

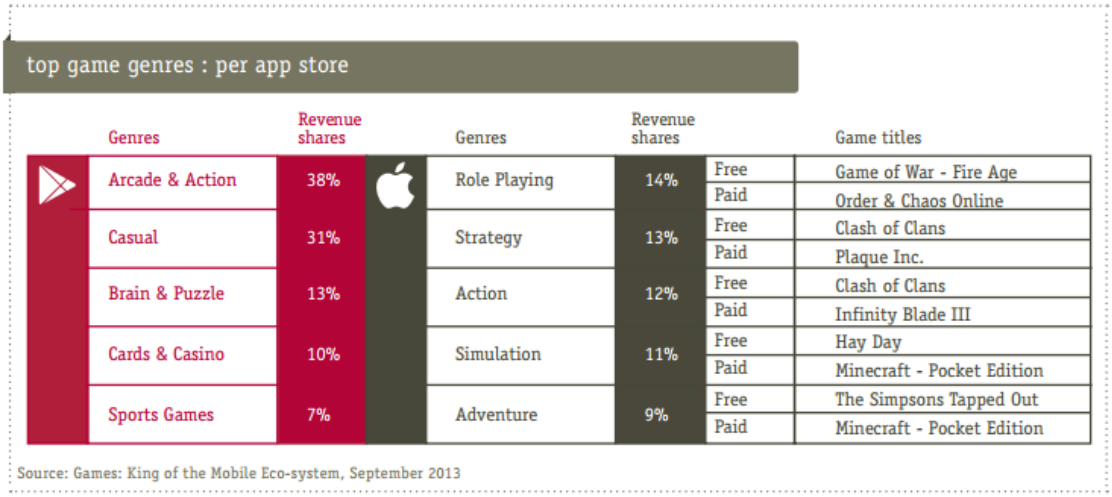


Figura 2.24: Popularidad de géneros en Google Play Store (izq) vs App Store de Apple (der) (2013)[20]

- **Su tienda oficial es la más adecuada para el modelo de negocio base escogido para el videojuego a desarrollar.** Atendiendo a las figuras 2.20 y 2.24, se puede observar como los poseedores de un dispositivo *Android* son los menos dispuestos a pagar por un videojuego de tipo casual. El modelo de negocio base escogido, distribución gratuita inicial con la posibilidad de adquirir contenido descargable opcional mediante de micro pagos, es el más adecuado para un videojuego con las características del videojuego a desarrollar.

- **La experiencia previa y los recursos iniciales.** Se dispone de experiencia previa en el desarrollo con *Android*, así como herramientas para desarrollar en sus plataformas (kits de desarrollo) y diversos terminales para poder realizar las pruebas que sean necesarias. Por el contrario, en el caso de *iOS*, no se posee ni de experiencia previa ni de ningún tipo de recursos iniciales para su desarrollo y pruebas.

Una vez establecido *Android*, como el sistema operativo que soporte el juego a desarrollar, es necesario determinar un rango de compatibilidad adecuado para el mismo. En otras palabras, determinar en qué versiones de *Android* es más adecuado centrarse en el desarrollo del videojuego. Esto no se puede determinarse con certeza hasta saber en cuál es el kit de desarrollo elegido para el desarrollo y sus capacidades. Esto será determinado en el análisis del estado del arte de la plataforma de desarrollo en el apartado siguiente.

[20] Fuente: <https://s3.amazonaws.com/CGA_Report/CGA_Market_Report_Fall2013.pdf>.

2.3.2 Análisis del kit de desarrollo de videojuegos

Partiendo del sistema elegido para el desarrollo en el apartado anterior, dispositivos móviles con sistema operativo *Android*, se procede a realizar un análisis del estado del arte de los kits o plataformas de desarrollo de videojuegos para dispositivos móviles *Android*. El objetivo de este análisis es determinar el kit de herramientas y/o la plataforma de desarrollo adecuado para desarrollar un videojuego con las características del videojuego a desarrollar, en el sistema elegido y con los recursos y experiencia iniciales.

Un kit de desarrollo de videojuegos es cualquier paquete de herramientas software o hardware diseñadas para ser utilizadas de forma conjunta para la creación de videojuegos. Entre las herramientas que lo forman se suelen encontrar: motores de videojuego y otros tipo de marcos de trabajo software, entornos de desarrollo integrados, programas de modelado 2d/3d, editores de sonido, etc.

En la actualidad, existen múltiples kits de desarrollo orientados para desarrollar videojuegos y aplicaciones en general para dispositivos móviles, ya sea exclusiva de un sistema operativo o *Cross-Platform*[21]. Para el sistema operativo *Android* en particular se pueden considerar los siguientes kits de herramientas o plataformas de desarrollo más relevantes en el mercado [36] [37] [38] [39] [40]:

Kit de desarrollo	Lenguaje	Orientación	Distribuición	Precio	Habilidad inicial
Unreal Engine 4	C++, C#, *UnrealScript* entre otros.	2D y 3D	*Android, iOS, HTML5, entre otras.*	19$/Mes + 5% de beneficio en bruto de cada videojuego.	Intermedia
Unity 3D	*C#, JavaScript*, *Boo* , otros lenguajes basados en .NET	2D y 3D	*Android, iOS*, *Windows Phone*, *BlackBerry 10*, entre otras.	Versión estándar gratuita. Versión Pro 1500 €	Intermedia
Marmalade	C++	2D y 3D	*Android*, *iOS, Blackburn entre otras.*	Versión básica: 149$. Versión estándar 499$	Intermedia
Project Anarchy	C++	2D y 3D	*Android*, *iOS*	Gratuito	Intermedia
GameMaker: Studio	Ninguno (Marco de trabajo gráfico)	2D	*Android*, *iOS*, *HTML5*	Versión *Windows* 39.99$. Versión *Mac* 39.99$ y Vresión *HTML5* 99$	Fácil
Corona Game Edition	Lua	2D y 3D	*Android*, *iOS*	Versión estándar 199$/Año. Versión Pro 349$/Año	Intermedia
Cocos2Dx	*Java*	2D	*Android*, *iOS*	Gratuito	Fácil

[21] Kit de desarrollo para dispositivos móviles que permite el desarrollo de aplicaciones para más de un sistema operativo o plataforma.

AppGameKit	*C++*	2D	*Android, iOS*	*Versión estándar 111.99$. Versión Pro 999$*	Intermedia
libgdx	*Java*	2D y 3D	*Android, HTML5*	Gratuito	Intermedia
AndEngine	*Java*	2D	*Android*	Gratuito	Intermedia
SDK de *Android*	*Java (Android)*	2D	*Android*	Gratuito	Difícil

Tabla 2.3: Características de los principales kits de desarrollo de videojuegos para Android

Determinar cuál de ellos es el más adecuado para el desarrollo de un videojuego con las características del videojuego a desarrollar no es tarea fácil, ya que muchos de los factores son subjetivos. Para llevar esto a cabo, se consideran las características principales de todos ellos, reflejadas en la tabla 2.3, la popularidad de cada uno, sus capacidades de cara al desarrollo y los recursos y la experiencia inicial que se dispone para el desarrollo. Igualmente se realiza un primer acercamiento con todos los kits que sean gratuitos (o que tengan una versión de demostración gratuita), para valorar de primera mano su funcionamiento y posibilidades.

Considerando todo lo anteriormente comentado, se determina que ***Unity 3D*** es el kit de desarrollo más adecuado para el desarrollo del videojuego objetivo. Esta decisión se ha tomado atendiendo los siguientes motivos, presentados en forma de los condicionantes más importantes que atañen al desarrollo:

Relación calidad – precio

La cuestión del precio es muy importante, ya que los recursos iniciales de los que se dispone para el desarrollo son limitados, obligando a descartar de antemano kits de desarrollo caros como *Marmalade* o *Corona*.

En el caso de Unity 3D, la relación calidad – precio es muy superior a las otras opciones. Su versión estándar gratuita tiene básicamente las mismas funcionalidades que su versión Pro de pago, a excepción de algunas opciones que ofrecen un control más profundo sobre los gráficos y la implementación en general (Ej. *Shaders* complejos) [41]. La falta de éstas no afecta de forma significativa al desarrollo y más tratándose de un videojuego no muy complejo para dispositivos móviles.

La calidad en general del kit de desarrollo al considerar su potencial y facilidad de uso, como se explicará más adelante, es excelente, sin nada que envidiar a otros kits de pago como *Marmalade* o *Corona*. Esto hace también que se descarten otras opciones gratuitas como *Project Anarchy* o *Cocos2Dx*.

Adicionalmente *Unity,* en su versión gratuita, no exige licencias, comisiones o *royalties* por publicar sus juegos en cualquiera de las plataformas que admite, siempre y cuando los beneficios (o presupuesto) de la entidad que lo esté utilizando no excedan los 100000 dólares anuales (en cuyo caso, no habría ningún impedimento en adquirir su versión Pro).

Relación facilidad de uso – potencial

La facilidad de uso al escoger un kit de desarrollo es un factor importante, ya que aspectos como la simplicidad de la IU, su flujo de trabajo, el lenguaje de programación que utilice o la forma que tenga de enmascarar su interacción con el propio SDK de *Android* son vitales a la hora de acelerar el tiempo de desarrollo. Es por ello que opciones con una dificultad ya de por sí alta, como programar directamente con el SDK de *Android*, se descartan.

En el caso de *Unity*, la IU (interfaz de usuario) del marco de trabajo principal es muy intuitiva, similar al de cualquier programa de composición de escenas 3d como *Blender* o *Maya*. La interfaz es modular y se organiza en distintos paneles de trabajo, de forma predeterminada (figura 2.25): visión de la escena del juego en la que se está trabajando, librería de *Assets* y panel de control de cada *Asset*. La comunicación entre los paneles se hace mediante el *drag and drop* de elementos de forma muy fluida e intuitiva.

El flujo de trabajo en *Unity* con el marco principal de trabajo, en cuanto a la construcción de escenas o niveles de los que se compone un videojuego, es bien sencillo: Primeramente se crean o se instancian los *Assets* que van a participar en la escena y se introducen dentro de la librería de *Assets*. Después estos *Assets* se introducen en la propia escena y se les asigna un comportamiento y características específicos mediante las opciones panel de control de los *Assets* o mediante scripts implementados en el entorno de desarrollo integrado de *Unity* (muy similar a otros entornos de desarrollo convencionales como *Eclipse*).

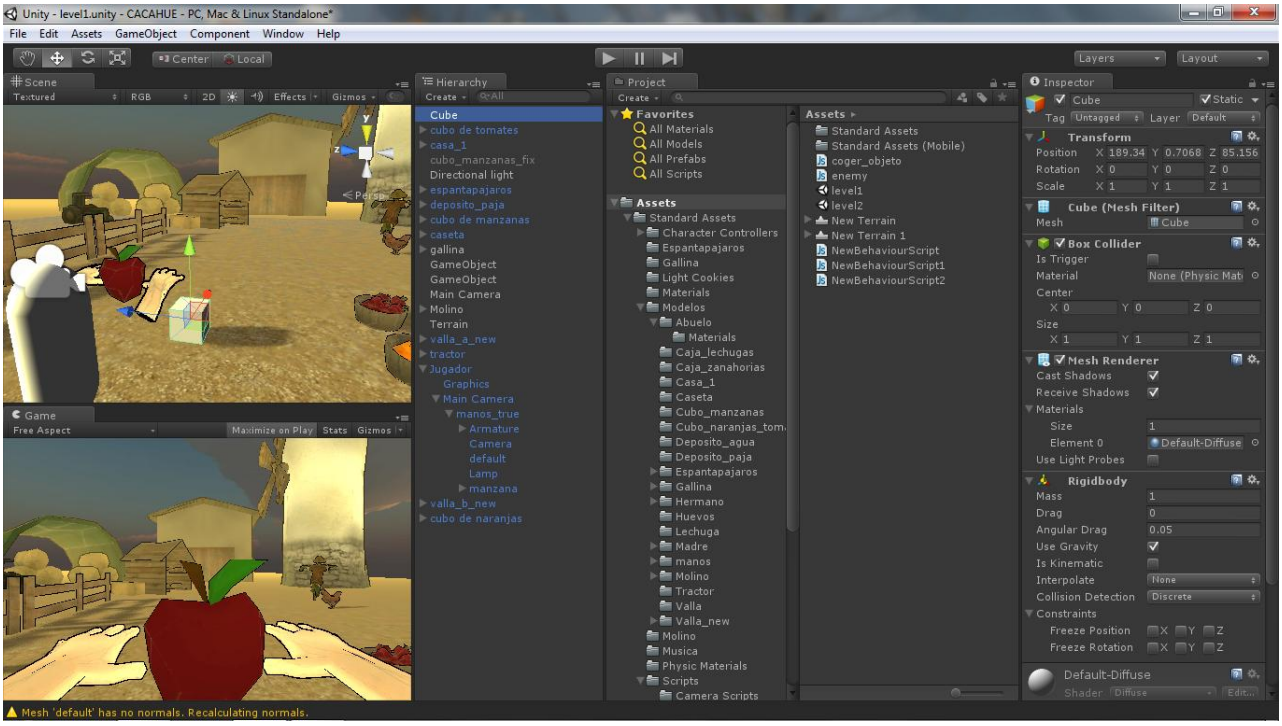


Figura 2.25: Marco de trabajo principal de Unity 3d

El lenguaje que se utiliza en *Unity* para la creación de scripts de comportamiento es *JavaScript* o *C#,* lenguajes que en muchos aspectos, requieren menos habilidad que los utilizados por otros kits de desarrollo alternativos, como C++ o *Lua*. Los scripts de comportamiento que se construyen con estos lenguajes tienen una estructura similar a las actividades de *Android*, con unos métodos que representan el ciclo de vida de un *Asset* y controlan *frame* a *frame* su comportamiento en la ejecución en tiempo real del juego (métodos como *Awaken(), Start(), Update(),* etc.).

A la hora de exportar el juego en distintas plataformas, *Unity* enmascara en gran medida la necesidad de utilizar código o *Assets* exclusivos para esa plataforma; a excepción de pequeños detalles, como por ejemplo ciertas funcionalidades que requieren una pantalla táctil u otro tipo de hardware exclusivo. Igualmente exime al desarrollador de preocuparse por temas como el rendimiento y la calidad gráfica del juego en la portabilidad. Esto hace que la portabilidad de *Unity* sea excelente en comparación con otros kits de desarrollo alternativos, sobre todo teniendo en cuenta la cantidad de plataformas a las que *Unity* permite exportar un mismo videojuego.

Por supuesto, toda esta facilidad de uso viene con un precio, ya que *Unity* no se salva de la dicotomía de que cuanto más fácil es utilizar un sistema de desarrollo, menos potencial tiene. Aspectos como el enmascaramiento y el lenguaje utilizado previenen que se pueda tener un control avanzado sobre los hilos en ejecución y la gestión de memoria en general, sobre todo al utilizar la versión estándar en vez la Pro. Esto afecta al rendimiento y a las capacidades gráficas, provocando que *Unity* no pueda llegar al mismo rendimiento y potencia gráfica que otros kits de desarrollo alternativos como *Unreal Engine 4.*

Aun así el potencial de *Unity* es más que suficiente para realizar juegos de aspecto profesional, en 2D y 3D, muy superiores a los desarrollados con otros kits alternativos también gratuitos (o de pago) y que requieren menor de habilidad por parte del usuario, como *GameMaker* o *Cocos2Dx*.

Popularidad y presencia en la comunidad

Debido a todas las virtudes comentadas anteriormente, *Unity* es el kit de desarrollo de videojuegos más popular globalmente, superando tres veces el porcentaje de uso de su competidor más cercano. Es el kit preferido por la mayor parte de la comunidad de desarrolladores de videojuegos, tanto amateur como profesionales [42].

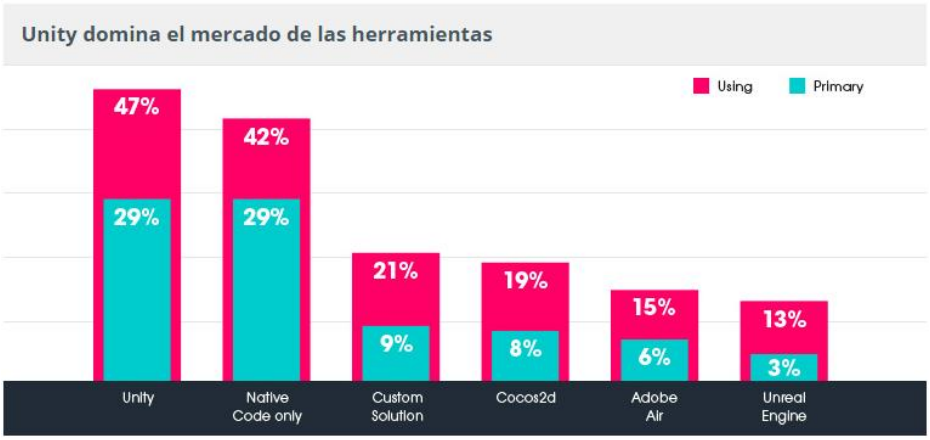


Figura 2.26: Presencia de Unity en el mercado global de los kits de desarrollo de videojuegos (2014) [22]

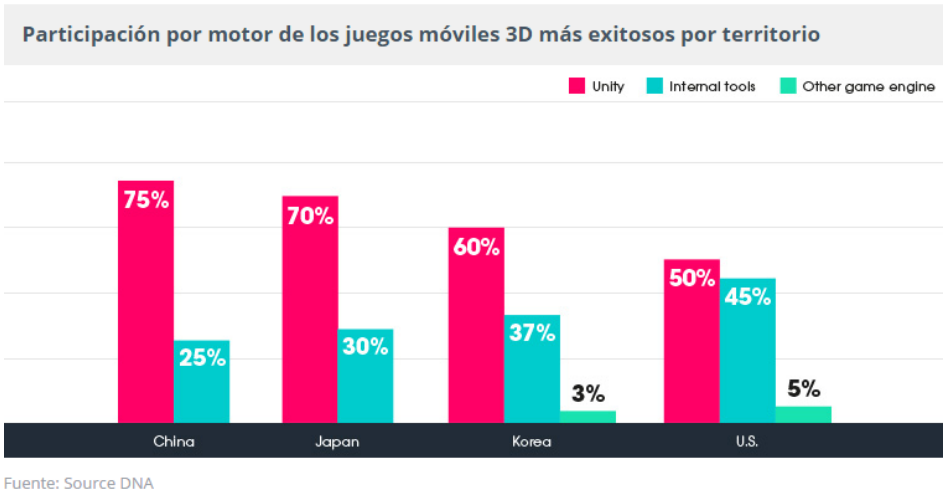


Figura 2.27: Participación de Unity en los juegos 3D de móvil más exitosos por territorio (2014) [23]

Por supuesto, toda la popularidad de *Unity* queda reflejada en su presencia en la comunidad de desarrolladores. Tanto de forma oficial mediante la documentación y el foro de *Unity* como por parte de foros o tutoriales de terceros, *Unity* recibe un soporte masivo por parte de la comunidad, mucho más que los otros kits alternativos. Esto sin lugar a dudas relaja la curva de aprendizaje de *Unity* de forma considerable y reduce los posibles problemas que se puedan tener durante el desarrollo.

Compatibilidad y líneas futuras

Como se ha comentado, a la hora de exportar un videojuego a distintas plataformas, *Unity* enmascara en gran medida la necesidad de utilizar código o *Assets* exclusivos para ellas. La mayor parte de los scripts de comportamiento se

[22] Fuente: <https://unity3d.com/es/public-relations>.
[23] Fuente: <https://unity3d.com/es/public-relations>.

implementan con la misma estructura común en *JavaScript* o C#, a excepción de algunos aspectos puntuales que están ligados al hardware o son intrínsecos del software de la plataforma de destino (ej. El mapeado de los botones del controlador de una plataforma en particular).

Esto hace que la portabilidad entre distintas plataformas sea muy fácil de llevar a cabo, logrando que por ejemplo un juego desarrollado para *Android* pueda ser portado a *iOS* sin apenas inconvenientes, sólo teniendo que cambiar unas pocas líneas de código.

Contando con las plataformas a las que *Unity* puede exportar aparte de *Android* (*iOS*, *Windows Phone*, *Playstation 3*, *Xbox 360*, etc.) y la gran portabilidad que presenta, es perfecto para la consideración de las líneas futuras del videojuego a desarrollar; ya que una vez desarrollado para *Android* podría ser portado a *iOS* o *Windows Phone* en el futuro sin apenas inconvenientes.

Por otro lado al considerar las capacidades de distribución de *Unity* respecto al sistema operativo *Android*, es posible determinar en qué versiones de *Android* es más adecuado centrarse en el desarrollo del videojuego. En este caso, *Unity* permite garantizar la compatibilidad del videojuego realizado con cualquier versión de *Android* que se encuentre a partir (mínimo) la versión 2.3.1 "*Gingerbread*".

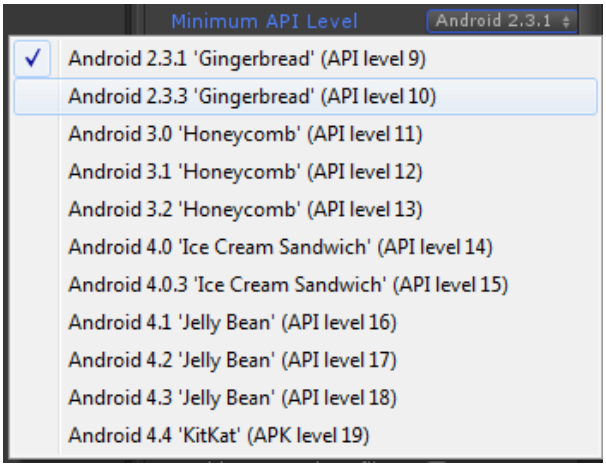


Figura 2.28: Selección de la versión mínima de compatibilidad de Android en Unity

Esto es más que suficiente para garantizar que el juego va a funcionar en un rango de compatibilidad de versión de *Android* aceptable [43], algo esencial teniendo en cuenta la disparidad de marcas de dispositivos y versiones de *Android* que existen en el mercado.

Experiencia inicial para el desarrollo

La experiencia inicial que se dispone para el desarrollo sobre los lenguajes de implementación que utiliza *Unity* (*Javascript* en concreto) es alta, en comparación con otros lenguajes utilizados por otros kits de desarrollo alternativos, como C++ o *Lua*, de los que se dispone una experiencia inicial más bien baja. Igualmente se dispone de

una mínima experiencia en cuanto a la navegación y el uso del *framework* principal de versiones más antiguas de *Unity*, que podría ayudar a acelerar el desarrollo.

A continuación se expone una tabla comparando de forma visual todos estos condicionantes para el desarrollo para cada uno de los kits de desarrollo de videojuegos analizados:

Kit de desarrollo	*Relación calidad - precio*	*Relación facilidad de uso - potencial*	*Popularidad y presencia en la comunidad*	*Compatibilidad y líneas futuras*	*Experiencia inicial para el desarrollo*
Unreal Engine 4	🔵	🟢	🟢	🟢	Ninguna
Unity 3D	🔵	🟢	🔵	🔵	Media
Marmalade	🟢	🟢	🟢	🟢	Ninguna
Project Anarchy	🟢	🟢	🟡	🟡	Ninguna
GameMaker: Studio	🟢	🟡	🟢	🟡	Media
Corona Game Edition	🟡	🟢	🟢	🟡	Ninguna
Cocos2Dx	🟢	🟡	🟡	🟡	Ninguna
AppGameKit	🟡	🟡	🟡	🟡	Ninguna
libgdx	🟢	🟢	🟡	🟡	Ninguna
AndEngine	🟢	🟡	🟡	🔴	Ninguna
SDK de *Android*	🟢	🔴	🟢	🔴	Media

Leyenda: 🔵 Especialmente buena; 🟢 Muy buena; 🟡 Poco buena; 🔴 Mala

Tabla 2.4: Comparativa de los principales kits de desarrollo de videojuegos para Android

2.3.3 Conclusiones del análisis

Teniendo en cuenta todo comentado en los análisis del estado del arte sobre el sistema y el kit de desarrollo a utilizar, atendiendo a todos los factores pertinentes, se considera como sistema que soporte al videojuego objetivo los **dispositivos móviles con sistema operativo *Android***. El kit de herramientas de desarrollo de videojuegos por su parte se considera el kit de desarrollo ***Unity 3d***.

3. Desarrollo

En este apartado se comentan las partes más relevantes del proceso de desarrollo del videojuego objetivo. Cubre las fases de **preproducción**, **producción** y **postproducción** de forma completa, a excepción de la sub-fase de pruebas (relegada a un apartado dedicado).

En la fase **preproducción** tienen lugar las sub-fases de concepción y diseño, donde se refinan las características base del videojuego (determinadas en el análisis del estado del arte) hasta dejarlo en condiciones para su implementación. Estas sub-fases culminan con la realización de una especificación de requisitos y un prototipo de bajo nivel de fidelidad que serán utilizados como guía durante la fase de producción.

En la fase de **producción**, se realiza la propia implementación del videojuego a desarrollar (codificación, creación de gráficos y sonido, etc.) a partir de las decisiones de diseño tomadas durante la preproducción y se comentan por encima los hitos o cambios respecto al diseño que hayan surgido. La fase culmina con un prototipo software de alto nivel de fidelidad, que será utilizado para la sub-fase pruebas.

Por último en la fase de **postproducción**, se determina el modelo de negocio final del videojuego a desarrollar, atendiendo al modelo de negocio base determinado en el análisis del estado del arte y a las características definitivas del prototipo de software de alta fidelidad. Además se propone un plan de gestión de actualizaciones y parches futuros una vez se haya distribuido el videojuego.

3.1 Preproducción – Concepción

En este apartado se comienza la fase de preproducción. Se establecen las ideas y conceptos necesarios para definir aun más el videojuego a partir de sus características base determinadas en el análisis del estado del arte. El objetivo es concretar las características base hasta dar con un concepto de juego adecuado para el diseño. **Se justifican aparte todas las decisiones tomadas en este proceso de refinamiento** de las características base originales.

3.1.1 Descripción general y características del videojuego

La descripción general del videojuego recoge todas las características base del mismo y las adapta para orientarlas mejor al propósito del juego y seguir una línea original de desarrollo. A continuación se muestra esta descripción a modo de ficha o sinopsis del videojuego, indicando con números partes de interés utilizadas en la justificación posterior:

- **Título del videojuego:** *CleanQuest* **(1)**
- **Tipo del videojuego:** Casual/Educativo.
- **Género del videojuego:** Compilación de mini-juegos con funcionalidades de realidad aumentada.
- **Número de jugadores:** 1. **(2)**
- **Edad recomendada:** de 6 a 9 años.

Descripción:

CleanQuest pone al jugador en la piel de Jaimito, el hijo más pequeño en una familia que vive en una casa rural con granja. El jugador, como Jaimito, deberá ayudar a distintos miembros de su familia a realizar tareas de limpieza de su casa en forma de divertidos mini-juegos; en el videojuego base: “Recoger los juguetes”, “Barrer el suelo” y “Limpiar los cristales” **(3)**.

Cada vez que el jugador completa una fase de un mini-juego, se le otorga un premio en forma de cromo virtual coleccionable, de una rareza equivalente a la habilidad que haya demostrado en el mini-juego. El jugador podrá disfrutar de la colección de cromos que haya ido obteniendo en la sección de “Galería de cromos” dentro del juego **(4)**.

Así mismo, en ciertos momentos, el videojuego se dirigirá al propio jugador para invitarle a participar en el “Desafío de realidad aumentada”. En este desafío el jugador deberá realizar las mismas tareas de limpieza del videojuego pero en la vida real, con el fin de obtener nuevos y raros cromos virtuales para añadir a la colección. La realización de las tareas en el desafío deberá ser confirmada por los padres o tutores del jugador, introduciendo un código específico en la sección “Galería de cromos”. El código respectivo de cada tarea deberá ser establecido la primera vez que se inicie el juego, por medio de un control parental **(5)**.

Justificación

(1) Título, trasfondo, dirección artística y jugabilidad – La justificación de los cambios realizados en estos aspectos a la hora de refinarlos se realizará en apartados dedicados más adelante.

(2) Nº de jugadores – La decisión para establecer el número de jugadores a 1 ha sido tomada en base a dos aspectos:

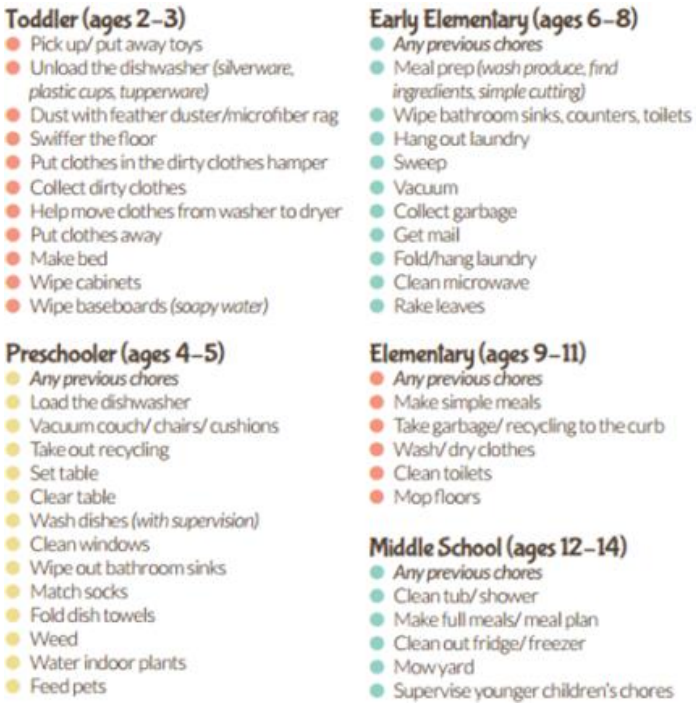
- Todos los videojuegos analizados en el estado del arte y la mayoría de los juegos casuales (educativos) en general, por su naturaleza, sólo soportan un jugador. Además la visión actual de los dispositivos móviles como asistentes personales, en vez de consolas de videojuegos, hace que la opción de sólo un jugador sea la más adecuada.

- La consideración de añadir al desarrollo un modo multijugador, ya sea offline u online, complicaría en exceso el desarrollo respecto a los recursos y experiencia iniciales que se disponen. El dedicar demasiado esfuerzo y recursos al aspecto multijugador en el desarrollo podría alejar al videojuego de su propósito inicial (herramienta para resolver el problema planteado). Por lo que considerar el videojuego para un solo jugador parece la opción más acertada.

(3) Nº de mini-juegos y selección de tareas de limpieza representativas – El número de mini-juegos que componen el juego ha sido reducido a 3: “Recoger los juguetes”, “Barrer el suelo” y “Limpiar los cristales”. Se ha estimado que 3 es un número suficiente para el juego base gratuito, con la posibilidad de añadir nuevos mediante micro pagos, como se apuntó en el modelo de negocio base.

Este número reducido aligera bastante la carga de desarrollo, permitiendo centrarse en mejorar la experiencia jugable de cada uno. En otras palabras, se apuesta por calidad antes que cantidad.

Las tareas que simulan cada uno han sido elegidas, por un lado, de acuerdo a las tareas más representativas que los niños del rango de edad objetivo (6 a 9 años) deberían ser capaces de realizar. Para ello se han cotejado diversos artículos y opiniones sobre la edad apropiada de cada una de las tareas del hogar (ej. Tabla de María Montessori [44]).



Toddler (ages 2–3)
- Pick up/ put away toys
- Unload the dishwasher *(silverware, plastic cups, tupperware)*
- Dust with feather duster/microfiber rag
- Swiffer the floor
- Put clothes in the dirty clothes hamper
- Collect dirty clothes
- Help move clothes from washer to dryer
- Put clothes away
- Make bed
- Wipe cabinets
- Wipe baseboards *(soapy water)*

Preschooler (ages 4–5)
- *Any previous chores*
- Load the dishwasher
- Vacuum couch/ chairs/ cushions
- Take out recycling
- Set table
- Clear table
- Wash dishes *(with supervision)*
- Clean windows
- Wipe out bathroom sinks
- Match socks
- Fold dish towels
- Weed
- Water indoor plants
- Feed pets

Early Elementary (ages 6–8)
- *Any previous chores*
- Meal prep *(wash produce, find ingredients, simple cutting)*
- Wipe bathroom sinks, counters, toilets
- Hang out laundry
- Sweep
- Vacuum
- Collect garbage
- Get mail
- Fold/hang laundry
- Clean microwave
- Rake leaves

Elementary (ages 9–11)
- *Any previous chores*
- Make simple meals
- Take garbage/ recycling to the curb
- Wash/ dry clothes
- Clean toilets
- Mop floors

Middle School (ages 12–14)
- *Any previous chores*
- Clean tub/ shower
- Make full meals/ meal plan
- Clean out fridge/ freezer
- Mow yard
- Supervise younger children's chores

Figura 3.1: Ejemplo de tabla de tareas apropiadas según edades[24]

Por otro lado, como se explicará más en detalle en el apartado "Jugabilidad y reglas" (apartado 3.1.4), dentro de las tareas apropiadas consideradas, se han escogido las más sencillas de simular con los mecanismos de jugabilidad esbozados en las características base durante el análisis del estado del arte.

(4) Característica de los cromos coleccionables – Como se comentó en el análisis del estado del arte de los productos de software que comparten objetivos de la solución, la premisa del juego a desarrollar se basa en dos principios para solucionar el problema planteado: Por un lado que el videojuego fomente las tareas de limpieza del hogar de forma positiva mediante la simulación y por otro que garantice la realización de estas tareas en la vida real mediante incentivos.

Para los incentivos se optó por utilizar elementos virtuales coleccionables, como ya hizo la plataforma *Choremonster*, que otorgaba al usuario monstruos virtuales coleccionables por completar tareas del hogar en la vida real aparte de los premios reales. Se decidió que estos incentivos fueran virtuales y coleccionables, ya que explotan la pasión de coleccionismo que ha llevado al éxito a otras franquicias infantiles como *Pokemon*.

Para el videojuego a desarrollar, *CleanQuest*, se establecen estos elementos virtuales coleccionables como cromos virtuales coleccionables. Estos cromos imitan a los tradicionales cromos coleccionables populares entre los niños (ej. Cromos de futbol, de coches, etc.) y su contenido está ambientado en el entorno de la granja

[24] Fuente: <http://thehappyhousewife.com/home-management/age-appropriate-chores-for-kids-printable/>.

donde vive Jaimito; pudiendo haber cromos que representen por ejemplo una gallina o un espantapájaros.

Como se indica en la descripción, estos cromos coleccionables podrán ser obtenidos al completar las fases de los mini-juegos o como incentivo para garantizar que el jugador realice las tareas de limpieza hogar en la vida real, a través del "Desafío de realidad aumentada". De esta manera, los cromos coleccionables se pueden considerar el nexo entre los dos principios que forman la premisa del videojuego, comentados anteriormente.

(5) Control parental y desafío de realidad aumentada – Para que los cromos virtuales coleccionables puedan cumplir su función como incentivo y el "Desafío de realidad aumentada" pueda funcionar, es necesario considerar algún tipo de sistema de control parental.

En el análisis del estado del arte de los productos de software que comparten objetivos de la solución (apartado 2.2), se comprobó como los productos tipo tablas de tareas *Choremonster* y *Family Chores*, presentaban un control parental de la siguiente forma: Los padres o tutores del usuario son los primeros en iniciar el servicio del producto, registrándose ellos y a sus hijos (usuarios) en el proceso y estableciendo las tareas que deben hacer cada uno en la vida real. Una vez que el usuario ha completado estas tareas, los padres o tutores deberán comprobar físicamente que la tarea se ha realizado para confirmar su realización en el producto y poder otorgar el incentivo o premio al usuario.

En *CleanQuest* se propone un sistema de control parental basado en el anterior: Al iniciar por primera vez el videojuego los padres o tutores del jugador (de forma análoga al registro anterior), deberán establecer un código o contraseña y un lugar físico para cada uno de los 3 mini-juegos que se encuentran en el juego base. Una vez introducidos, el jugador ya podrá disfrutar del videojuego. Más adelante, cuando el jugador complete todas las fases de un mini-juego, el videojuego le invitará a participar en el "Desafío de realidad aumentada". En este desafío se le indicará que, para ganar un nuevo y raro cromo coleccionable, deberá realizar la tarea de limpieza análoga del mini-juego pero en la vida real y en un lugar (usualmente de su casa) determinado y al finalizarlo, pedirle a sus padres o tutores que introduzcan el código o contraseña correspondiente en la sección de "Galería de cromos" y desbloquear así ese nuevo y raro cromo.

Cabe destacar que en el contexto de *CleanQuest*, se considera como "realidad aumentada" cualquier interacción que tenga el videojuego con la realidad que no sea mediante sus controles principales táctiles (su interfaz). Debido a esto se decidió llamar al "Desafío de realidad aumentada" como tal, ya que mediante el mismo (incluyendo el proceso de introducción de los códigos/contraseñas de confirmación), el videojuego era capaz de reconocer la realización de tareas de limpieza en la vida real por parte del jugador.

De esta forma, una vez que el jugador efectivamente completa esa tarea en la vida real, podrá pedirle a sus padres o tutores el código para desbloquear el cromo. Los padres por su parte (de forma análoga a la confirmación que se realizaba en

Choremonster y *Family Chores*) tendrán que comprobar físicamente que el jugador a realizado la tarea en el lugar determinado antes de introducir el código. Así se garantiza que mediante el control parental y el incentivo de los cromos virtuales coleccionables el jugador verdaderamente realizará las tareas de limpieza del hogar en la vida real.

3.1.2 Título del videojuego

El título propuesto para el videojuego a desarrollar, objeto de este TFG, es ***CleanQuest***, nombre por el cual se referirá al mismo durante el resto de esta memoria.

Justificación

El título *CleanQuest*, está compuesto por dos palabras en inglés: *Clean*, limpio y *Quest*, búsqueda o aventura. El título forma parte del trasfondo, ya que, como se explicará en el apartado dedicado más adelante, las fases de cada mini-juego y el juego en general transcurren en distintos momentos de la vida diaria del protagonista, Jaimito, en los cuales tiene que realizar varias tareas de limpieza del hogar. De esta forma, para Jaimito y por consecuencia para el jugador, su vida diaria se convierte en una "búsqueda o aventura de la limpieza". El título está pensado para ser llamativo e involucrar al jugador en la solución del problema, como el resto del trasfondo y la dirección artística.

3.1.3 Estructura y flujo general del videojuego

La estructura y el flujo conceptual de *CleanQuest*, considerando las pantallas o secciones principales de las que va a disponer, es la siguiente:

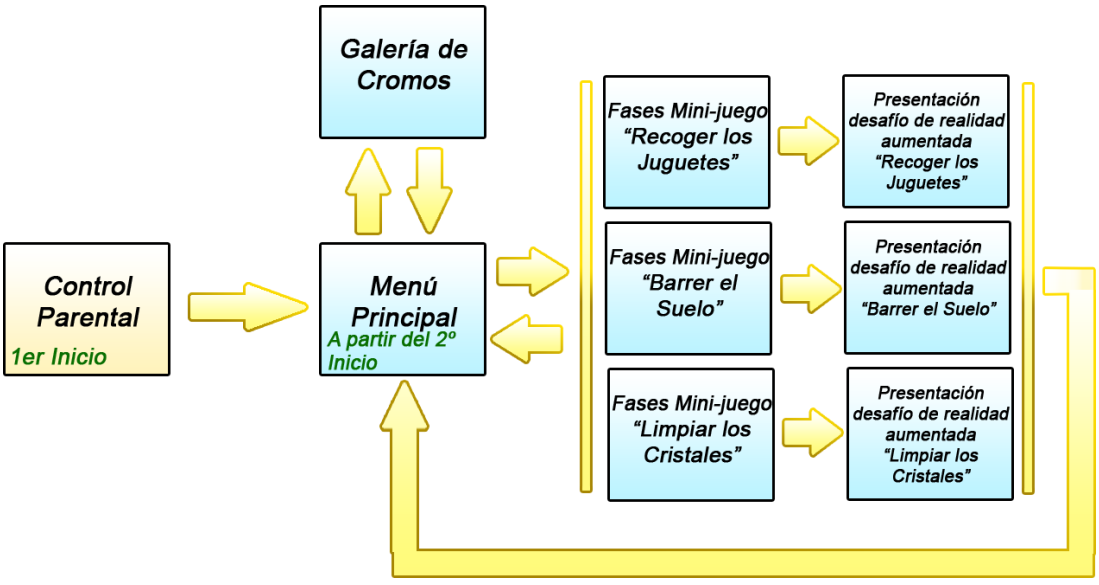


Figura 3.2: Estructura y flujo conceptual de CleanQuest

Justificación

La estructura y el flujo de *CleanQuest* son similares a los de muchos videojuegos convencionales, pero simplificados para que puedan ser los suficientemente intuitivos para que los jugadores del rango de edad objetivo y los usuarios de dispositivos móviles en general puedan reconocer el propósito de cada pantalla fácilmente y navegar entre ellas sin problemas.

Desde el menú principal, el jugador puede acceder a la "Galería de cromos virtuales coleccionables" o a cualquier fase de un mini-juego que previamente haya desbloqueado. Igualmente, el jugador puede regresar al menú principal desde estas pantallas en cualquier momento. Cuando un jugador completa todas las fases de un mini-juego, accederá a la presentación del "Desafío de realidad aumentada" de la tarea que represente el mini-juego; aquí el videojuego le indicará en qué consiste el desafío y lo que tiene que hacer para superarlo (como se comentó en el apartado de descripción general y características del videojuego).

Lo único destacable, que difiere de la mayoría de los videojuegos convencionales, es el control parental. La primera vez que se inicie el videojuego, deberán hacerlo los padres y tutores, introduciendo los códigos y contraseñas para cada mini-juego con el fin de que el "Desafío de realidad aumentada pueda funcionar". Una vez que se hayan introducido, las subsiguientes veces que se inicie el videojuego, no será necesario volverlas a introducir y comenzará directamente en el menú principal.

3.1.4 Jugabilidad y reglas

La jugabilidad y reglas en los mini-juegos de *CleanQuest,* se proponen de la siguiente forma, desglosándolas de forma general y para cada mini-juego:

- **En general, para todos los mini-juegos.** Todos los mini-juegos siguen el mismo reglamento genérico, que afectará a la jugabilidad. El objetivo de cada fase en un mini-juego es completar una serie de hitos o acciones (que puedan ser numeradas y contabilizadas) lo más rápidamente posible, ya que habrá un contador de tiempo progresivo que determinará al final de la fase el tiempo que ha tardado en completarla. Cuanto menos tiempo haya tardado el jugador en completar la fase, se le otorgará un premio equivalente a un cromo coleccionable de mayor valor, aunque, como el contador progresivo es indefinido, siempre se podrá optar al premio de menor valor (nunca se termina una fase sin un premio). Cada mini-juego estará compuestos de 3 fases, cada una más difícil que la anterior (los cromos de mayor valor tendrán un margen de tiempo para completar la fase más corto u otras variables propias de la naturaleza del mini-juego que añadan dificultad a la fase).

- **Jugabilidad del mini-juego "Recoger los juguetes"**. Mini-juego en perspectiva de primera persona en el que el jugador debe recoger los juguetes en la habitación de Jaimito. La acción/hito numerable que el jugador debe realizar es coger un juguete y meterlo en un baúl mediante un mecanismo simple tipo *drag and drop.* En cada fase posterior a la primera, se elevará la dificultad para conseguir cromos de mayor

rareza, disminuyendo el margen de tiempo para conseguirlos o incrementando la cantidad de acciones/hitos necesarios para completar la fase.

- **Jugabilidad del mini-juego "Barrer el suelo"**. Mini-juego en perspectiva de tercera persona en el que el jugador debe barrer el suelo del salón de la casa de Jaimito. La acción/hito numerable que el jugador debe realizar es conducir una escoba por el escenario barriendo unidades de suciedad (pelusas) hasta situar alguna de ellas en una zona determinada (un rincón). El manejo de la escoba se realiza mediante un *joystick* táctil embebido en la pantalla, un botón para agitar la cabeza de la escoba (propinar un "escobazo" para desplazar con más ímpetu las pelusas) y otro botón auxiliar para cambiar la dirección a la que apunte la escoba. En cada fase posterior a la primera, se elevará la dificultad para conseguir cromos de mayor rareza, disminuyendo el margen de tiempo para conseguirlos o incrementando la cantidad de acciones/hitos necesarios para completar la fase.

- **Jugabilidad del mini-juego "Limpiar los cristales"**. Mini-juego en perspectiva de primera persona en el que el jugador debe limpiar las ventanas de la granja de Jaimito. La acción/hito numerable que el jugador debe realizar es limpiar con un trapo una mancha de la ventana de las muchas que hay. El manejo del trapo se realiza mediante *drag and drop,* cogiendo el trapo sin soltarlo y restregándolo contra las manchas hasta hacerlas desvanecer completamente. En cada fase posterior a la primera, se elevará la dificultad para conseguir cromos de mayor rareza, disminuyendo el margen de tiempo para conseguirlos o incrementando la cantidad de acciones/hitos necesarios para completar la fase.

Justificación

La jugabilidad de *CleanQuest,* como se comentó en las características base durante el análisis del estado del arte de los productos de software que comparten objetivos de la solución (apartado 2.2), deberá estar basada en mecanismos simples e intuitivos estilo *drag and drop* y pulsaciones en la pantalla táctil. Estos mecanismos son utilizados conjuntamente de forma creativa para que, mediante ellos, se puedan imitar las mismas acciones que se realizarían en la vida real para estas tareas de limpieza. El objetivo de esto es que el videojuego sea lo suficientemente intuitivito para que los niños del rango de edad indicado, al encontrarse en la escena de un mini-juego, sepan sin necesidad de ayudas o indicaciones como desenvolverse con los controles.

Los mini-juegos propuestos presentan una jugabilidad que usa estos mecanismos de una forma acertada, similar a los productos de software analizados en el estado del arte de tipo simulación. Las tareas de limpieza a las que representan han sido escogidas atendiendo, entre otros aspectos (como el rango de edad apropiado), a la facilidad que tienen sus formas de realización para ser imitadas o simuladas con estos mecanismos.

La única excepción es el mini-juego de "Barrer el suelo" en el que se ha decidido aplicar una jugabilidad algo más compleja y desafiante, dirigida a los jugadores más mayores dentro del rango de edad objetivo: niños de 9 años. Este tipo de jugadores pueden encontrar la jugabilidad del resto de los mini-juegos demasiado simple, por lo

que es conveniente incluir algún mini-juego con un requerimiento de habilidad más elevado que la media. Se ha elegido este mini-juego en concreto ya que la tarea de "Barrer el suelo" es más indicada para niños de 9 años, según multitud de "tablas de tareas apropiadas según edad" como la de la figura 3.1.

Las reglas comunes de todos los mini-juegos han sido pensadas para dar más profundidad a la jugabilidad y mantenerlos atractivos a los usuarios, como ya hizo *Kids House Clean Up*. Se utiliza una penalización mínima, ya que el usuario siempre podrá optar al premio de menor valor, para evitar el rechazo que podría surgir de los usuarios con menor habilidad. En otras palabras, el objetivo es que el jugador siga encontrando cierto desafío incluso en el mismo mini-juego, intentando superar su puntuación (en este caso, completar la misma fase en un tiempo menor) y así optar a nuevos premios en forma de cormos coleccionables.

3.1.5 Dirección artística

La dirección artística que se propone para *CleanQuest*, considerando gráficos, animaciones y sonido, es la siguiente:

- **Gráficos.** *CleanQuest* presenta gráficos en 3d, con una interfaz de usuario (botones, menús, etc.) en 2d. El estilo de los gráficos 3d es simple, con modelos *low poly*[25], caricaturizados (con un diseño original y desenfadado) y utilizando técnicas de *cell shading*[26]; otorgándoles de esta manera un aspecto vistoso, colorido y reconocible, con una ambientación única, imitando a los dibujos animados. Las interfaces 2d por su parte, presentan un estilo sencillo, colorido y vectorial, acorde con los gráficos 3d.

Figura 3.3: Ejemplo de gráficos 3d con estilo cell shading[27]

[25] Estilo de modelado 3d que consiste en hacer modelos con una carga de polígonos considerablemente baja.

[26] Estilo visual de gráficos 3d que hace parecer a los modelos rodeados por una línea continua gruesa, simulando el aspecto de dibujos animados.

[27] Fuente: <http://www.ozom.cl/unreal-engine-4-unity-5-y-cryengine-los-3-motores-graficos-que-compiten-por-la-supremacia/>.

- **Animaciones.** Las animaciones tanto de personajes, como objetos y de la interfaz (ej. paso entre menús) son simples y fluidas, complementando el estilo de los gráficos.

- **Sonido.** La música y los efectos de sonido son acordes al estilo del juego en general: simples, desenfadados y divertidos. La música de cada pantalla o menú contribuye a la ambientación general de la escena.

Justificación

El hecho de que los gráficos sean 3d, es debido a que *Unity*, aunque sea capaz de producir gráficos en 2d y 3d, está más adaptado para 3d y tiene más soporte para ello. De esta forma se facilitaría en gran medida el desarrollo. Por otro lado, los gráficos 3d están de moda [45] en toda la industria cinematográfica y de animación infantil en general, por lo que esta decisión podría aumentar el atractivo del videojuego hacia el usuario objetivo. Se ha decidido por una interfaz de usuario 2d convencional, ya que para cumplir su función es suficiente y una interfaz 3d, aunque más vistosa, complicaría innecesariamente el desarrollo

El estilo de los gráficos 3d, *cell shading*, *low poly* y de diseño caricaturesco, como si fuera un dibujo animado, atiende a los siguientes motivos: Por un lado, las combinación de las técnicas de *low poly* y *cell shading* producen unos modelos 3d de baja carga poligonal y con unas texturas simples y de baja resolución, altamente recomendables si el sistema que va a soportar el videojuego dispone de relativamente poca potencia gráfica o memoria, como es el caso de los dispositivos móviles. Por otro lado, el aspecto resultante de la utilización de estas técnicas, de dibujo animado simple, colorido y llamativo, aumenta en gran medida el atractivo del videojuego hacia los usuarios del rango de edad objetivo.

Todo el diseño gráfico en general, la animación, la música y los efectos de sonido, están pensados para hacer más atractivo el videojuego hacia el tipo de jugador objetivo y contribuir de forma indirecta al trasfondo.

3.1.6 Trasfondo

El trasfondo de *CleanQuest* es el siguiente, desglosado en cuanto a la forma que tiene de llegar al jugador e indicando con números partes de interés que serán utilizadas en la justificación posterior:

- **De forma directa.** *CleanQuest* pone al jugador en la piel de Jaimito **(1)**, el hijo más pequeño de una familia que vive en una casa rural que cuenta con una granja **(2)**. El jugador, como Jaimito, tiene que ayudar a distintos miembros de su familia **(3)** a realizar tareas de limpieza de su casa en forma de divertidos mini-juegos. En cada mini-juego, Jaimito deberá ayudar a un miembro de la familia distinto: su madre en el de “Recoger los juguetes”, su hermano mayor en el de “Barrer el suelo” y a su abuelo en el de “Limpiar los cristales”. Cada mini-juego cuenta con 3 fases, situadas temporalmente en 3 partes de un mismo día (mañana, tarde y noche) **(4)**. En el inicio de cada fase, el miembro de la familia correspondiente a ese mini-juego,

explicará a Jaimito la razón por la cual tiene que ayudarle a hacer la tarea de limpieza, a través de una escena cinemática o *cutscene*[28] (ej. La madre de Jaimito le pide que recoja los juguetes en su habitación ya que van a venir sus tíos de visita y quiere ver la casa recogida). Igualmente se utilizarán *cutscenes* para, al final de cada fase, presentar los premios y cromos coleccionables que ha conseguido el jugador **(5)**. En cada *cutscene* de inicio de fase, el miembro de la familia correspondiente podrá mencionar o retomar situaciones de fases pasadas o de incluso otros mini-juegos **(6)**.

- **De forma indirecta.** Todos los elementos de *CleanQuest* (escenarios, personajes, objetos, etc.) forma parte del trasfondo general en mayor o menor medida. Cada miembro de la familia de Jaimito está caracterizado de una forma única y original que demuestra en cada *cutscene* de inicio y final de fase; tanto su apariencia, como su personalidad y forma de hablar aportan pedazos de información sobre el trasfondo al jugador. Los escenarios donde se desarrolla la acción presentan múltiples detalles (ej. Por una ventana se puede ver a otros miembros de la familia trabajando en la granja) que colaboran con el trasfondo de forma indirecta. Igualmente, en la sección de "Galería de cromos", al examinar cada cromo coleccionable conseguido, se podrán descubrir más detalles sobre los personajes u objetos del entorno que representan esos cromos **(7)**.

Justificación

(1) Jugador como Jaimito – Se ha decidido que el jugador encarne a Jaimito en *CleanQuest* debido a que prácticamente todos los productos de software analizados de tipo simulación durante el análisis del estado del arte presentaban esta característica. Videojuegos como *Abby's Home Laundry* o *Kids House Clean Up* ponen al jugador en la piel de un avatar que encaja dentro de su trasfondo.

El objetivo de esto, como el de la mayoría del trasfondo en general, es que el jugador se involucre y se introduzca dentro del hilo del juego, considerando las tareas de limpieza que realiza como una responsabilidad que le han asignado a Jaimito y que sólo está en la mano de los jugadores ayudarle en estas tareas.

El hecho de que se haya elegido un protagonista masculino, es para evitar promocionar los roles de género en la sociedad como ya hicieron en juegos como *Baby Home Adventure*, en el que todos los avatares que te dejaba seleccionar eran femeninos.

(2) Escenario/Ambientación general como una casa rural con granja – Se ha decidido utilizar este escenario en particular, en vez de otro más convencional, como un aliciente más para aumentar el atractivo hacia los jugadores objetivo y darle un enfoque más original a la dirección artística.

[28] En el contexto de los videojuegos una *cutscene* es una secuencia cinemática en el que el jugador no tiene control o lo tiene de forma limitada. Se utilizan para avanzar la trama, explicar alguna información útil para el jugador, presentar información de fondo, etc.

Una granja es un entorno comúnmente relacionado positivamente con la infancia, y que gusta a los niños en general. Esto puede comprobarse en los colegios de primaria, donde muchas de las excursiones que organizan es hacia granjas o en muchos zoos donde es típico encontrarse una zona llena de animales de granja sólo para niños pequeños.

(3) Personajes no jugables que encargan las tareas de limpieza al jugador – La inclusión de los miembros de la familia de Jaimito como personajes no jugables que encargan las tareas de limpieza en cada mini-juego, aparte de enriquecer el trasfondo aumentando la inmersión del jugador y su involucración en las tareas, promueven el respeto y la empatía a la figura autoritaria de los miembros más mayores de la familia. Estos personajes, vistos como figuras autoritarias, aumentan la responsabilidad del jugador y fomentan de forma positiva las tareas de limpieza que representan los mini-juegos, ya que el jugador verá que la realización de estas tareas es algo necesario, que haría feliz a estos personajes (incluyendo al propio Jaimito) dentro del entorno doméstico.

(4) y (6) Reflejo del paso del tiempo en *CleanQuest* – El hecho de que cada fase represente distintas partes de un mismo día y que los personajes no jugables puedan referenciar situaciones pasadas o futuras, simplemente está para aumentar la inmersión del jugador en el trasfondo y que vea que sus acciones verdaderamente tienen un impacto en el entorno del juego al pasar el tiempo.

(5) Uso de *cutscenes* – El recurso de las *cutscenes* es una técnica muy común en la industria del videojuego. En el caso de *CleanQuest*, estas *cutscenes* se utilizan al inicio de cada una de las fases de un mini-juego, para que los miembros de la familia de Jaimito (personajes no jugables) informen al jugador de lo que tiene que hacer en el mini-juego (en qué consiste) y sobre otros datos del trasfondo.

El objetivo de estas *cutscenes* es:

- Aumentar el atractivo del videojuego hacia el jugador, dándole un aspecto más vivo y creíble, en contraste de las introducciones con muros de texto de juegos como *Abby's Home Laundry*.
- Enriquecer el trasfondo general, utilizándolas como vía para mostrar la caracterización y personalidad de los miembros de la familia de Jaimito, así como otros detalles del trasfondo.
- Aumentar el efecto que producen los miembros de la familia de Jaimito como figuras autoritarias y el fomento de las tareas de limpieza que encargan de forma positiva como se ha comentado en el punto (3).

(7) Presentación del trasfondo de forma indirecta mediante la dirección artística – La dirección artística de *CleanQuest* contribuye de forma indirecta a enriquecer el trasfondo del videojuego con el objetivo de aumentar su atractivo hacia el usuario objetivo de forma subliminal y fomentar positivamente las tareas de limpieza que representan los mini-juegos. De igual forma, el trasfondo que comunican de forma indirecta los cromos coleccionables también contribuye a aumentar su atractivo como elemento coleccionable y aumenta sus posibilidades de éxito como incentivos.

3.2 Preproducción – Diseño conceptual y artístico

La fase de diseño continúa la fase de concepción. Se parte de las ideas concebidas en esa fase (que a su vez partieron de las características base halladas en el análisis del estado del arte) y se definen aun más, se organizan y plantean. Concretamente, esta sub-fase de diseño se centra en las **ideas que necesitan un grado de detalle más alto respecto a su faceta conceptual y artística**, con el fin de **optimizar el videojuego como herramienta para solucionar el problema planteado**. Esto es necesario antes de poder continuar con la fase de diseño de más peso en el desarrollo: El diseño de implementación y técnico (apartado 3.4). Esto incluye el diseño de personajes no jugables; diseño de objetos y escenarios; diseño de sonido; diseño del guión; diseño de la interfaz de usuario; y diseño del control parental, cromos coleccionables y dinámica de premios.

Cabe destacar que para este tipo de diseño se ha tenido en cuenta la "*Ley de Propiedad Intelectual*" (modificada y aprobada por el "*Real Decreto Legislativo 1/1996*") [46] en lo que respecta al contenido no original. Para el diseño del videojuego a desarrollar se utiliza sólo contenido original o contenido obtenido de repositorios sin derechos de autor (que permitan su comercialización).

3.2.1 Diseño de personajes no jugables

Como se comentó en el apartado de "Preproducción – Concepción" (apartado 3.1) se ha decidido incluir en *CleanQuest* 3 personajes no jugables, relevantes de cara al desarrollo. Estos personajes son miembros de la familia de Jaimito y cada uno ha sido asignado/asiste a un mini-juego distinto: su madre en "Recoger los juguetes", su hermano mayor en "Barrer el suelo" y su abuelo en "Limpiar los cristales". Sus competencias dentro del videojuego son:

- Explicar a Jaimito como tiene que abordar cada mini-juego (su objetivo y lo que tiene que hacer para alcanzarlo) de forma sutil, intercalando fragmentos del trasfondo. "Sutil" ya que no debería ser necesario describir los controles y reglas del mini-juego de forma explícita; se esperan que sean lo suficientemente intuitivos para ello. Esto ocurre al inicio de cada fase en cada mini-juego a través de una *cutscene*.

- Felicitar a Jaimito e informarle sobre cada premio obtenido al final de cada fase, en cada mini-juego, a través de una *cutscene.*

- Una vez completadas todas las fases de un mini-juego, informar al propio jugador (no al avatar de Jaimito que controla el jugador) mediante una *cutscene* sobre el "Desafío de realidad aumentada" referente a la tarea de limpieza que representa ese mini-juego e invitarle a participar en él.

– Formar parte del trasfondo, de forma directa, al comentar sucesos o anécdotas de su entorno en las *cutscenes* o de forma indirecta, a través de su personalidad y caracterización o formando parte del decorado del escenario en ciertos mini-juegos.

Comprobada la importancia de estos personajes dentro del juego, es necesario reservar este apartado dedicado a ellos, donde se refleje el proceso de su diseño. El objetivo de este proceso de diseño es aplicar la dirección artística planteada en apartados anteriores, de tal forma que los personajes no jugables sean aptos para cumplir todas las competencias anteriormente comentadas.

El proceso de diseño se refleja de forma desglosada, mostrando sus partes más relevantes referentes a la descripción general, apariencia gráfica y animaciones de los personajes no jugables:

Descripción general de los personajes no jugables

Breve descripción de cada personaje no jugable, basada en el trasfondo, que servirá de apoyo para crear su apariencia gráfica y animaciones e interpretar su voz en cada *cutscene*.

- **PNJ "Madre de Jaimito" –** "*Mujer de mediana edad, caucásica con ojos marrones, pelo largo marrón, alta y de complexión delgada. Va vestida con una ropa típica de trabajo en la granja: camiseta blanca, un peto de trabajo vaquero y guantes de trabajo. Presenta un carácter afable pero estricto hacia su hijo*."

- **PNJ "Hermano mayor de Jaimito" –** "*Hombre adolescente, caucásico con ojos naranjas, pelo corto naranja de punta, alto y de complexión delgada. Va vestido con una camiseta sin mangas negra con el símbolo de la paz impreso en ella y pantalones vaqueros. Presenta un carácter afable pero algo inseguro hacia su hermano*."

- **PNJ "Abuelo de Jaimito" –** "*Hombre viejo, caucásico con ojos grises; pelo corto, bigote, cejas pobladas y barba larga también grises, alto, encorvado y de complexión delgada. Va vestido con un peto vaquero de trabajo en la granja sin camiseta, guantes de trabajo y un sombrero de paja en la cabeza. Lleva siempre consigo un viejo horquillo para trabajar. Presenta un carácter divertido, algo senil y despreocupado hacia su nieto*."

Apariencia de los personajes no jugables

El diseño de la apariencia gráfica de los personajes no jugables está sujeto, por un lado, a la dirección artística especificada en la fase de concepción (modelos simples, *low poly*, técnicas *cell shading*, etc.) y por otro a las descripciones anteriores.

Para llevar a cabo este diseño se realizan bocetos de arte conceptual (utilizando el programa *Photoshop*) de los distintos aspectos de la apariencia de cada personaje no jugable. Éstos se utilizarán como referencia a la hora de realizar los modelos 3d pertinentes en la fase de producción.

A continuación se muestran algunos de los bocetos más relevantes realizados durante esta fase de diseño que muestran la apariencia de cada personaje no jugable, de izquierda a derecha: La madre, el hermano y el abuelo de Jaimito.

Figura 3.4: Muestra de arte conceptual de los PNJ de CleanQuest

Animaciones de los personajes no jugables

Al considerar las animaciones para todos los elementos (modelos 3d e interfaces) del juego, éstas se pueden clasificar en dos grupos: Animaciones implementadas de forma programática mediante *Unity* y animaciones ligadas al modelo, realizadas con un programa externo. Las animaciones de los personajes no jugables, debido a su complejidad (integración de un esqueleto virtual o *rig*, movimiento coordinado de distintas extremidades, movimiento de la boca al hablar, etc.), pertenecen al segundo grupo.

Para no complicar en exceso el desarrollo, considerando los recursos y la experiencia iniciales relativos a la animación de modelos 3d, se decide simplificar la animación de los personajes no jugables siguiendo estos 3 principios:

- Reducir al mínimo el número de animaciones por modelo, estableciendo un conjunto de animaciones esenciales para todos los modelos (ej. Conjunto de acciones: “Hablar”, “Felicitar” y “Esperar”).
- Simplificar cada una de las animaciones lo más posible; integrando un esqueleto minimalista en los modelos, animando solo de cuerpo para arriba,

ignorando la mayoría de animaciones faciales, utilizando una animación neutra en la boca a la hora de hablar, etc.

- Construir animaciones más complejas dentro de *Unity* de forma programática, combinando animaciones simples o reproduciéndolas con bucles.

3.2.2 Diseño de objetos y escenarios

Aparte de los personajes no jugables, *CleanQuest* cuenta con una serie de elementos a considerar (no referentes a la interfaz) que representan objetos y escenarios dentro del juego y que deben pasar por un proceso de diseño similar al de los personajes no jugables, ya que no dejan de ser otros modelos 3d. Estos elementos pueden clasificarse en los siguientes grupos, atendiendo a su competencia dentro del juego:

- **Escenarios –** Los escenarios son los modelos 3d "contenedores" donde se desarrolla la acción en cada mini-juego. Como ya se comentó en la fase de concepción, se proponen 3 escenarios globales, uno por mini-juego: La habitación de Jaimito para el mini-juego "Recoger los juguetes", el salón de Jaimito para el mini-juego "Barrer el suelo" y el establo/granero de la granja de Jaimito para el mini-juego "Limpiar los cristales". Aunque el escenario global para cada mini-juego se podría considerar como el conjunto de modelos 3d del escenario más todos los objetos (decorativos, pasivos y activos) que contiene, para esta clasificación sólo se considera el esqueleto del mismo, esto es, el propio habitáculo (4 paredes, techo y suelo) más todos sus accesorios (puertas, ventanas, cuadros colgados, etc.). Debido a que este esqueleto puede tener impacto sobre la jugabilidad (ej. Alguna de sus paredes haciendo de límite para algún tipo de interacción), los escenarios también son considerados objetos que afectan a la jugabilidad de forma pasiva.

- **Objetos decorativos –** Modelos 3d que no repercuten de ninguna manera en la jugabilidad. Complementan a los escenarios y les añaden ambientación y trasfondo. Entre ellos destacan los objetos que se encuentran en el exterior de los escenarios como otras edificaciones de la granja, personas, animales o el propio terreno que pueda verse a través de las ventanas.

- **Objetos que afectan a la jugabilidad de forma pasiva –** Modelos 3d que afectan a la jugabilidad de forma pasiva, esto es, estableciendo límites, barreras u obstáculos físicos para la interacción con los objetos que afectan a la jugabilidad de forma activa. Su objetivo es regular la dificultad de la fase donde se encuentren respecto al objetivo de ese mini-juego. Ejemplos de estos objetos se pueden considerar. en el mini-juego de "Barrer el suelo", un mueble que estorbe para barrer las pelusas hasta su rincón; en el mini-juego de "Recoger los juguetes", una silla que debajo tenga un juguete y el jugador tenga que sacarlo de entre sus patas para poder recogerlo; o el propio escenario del mini-juego.

- **Objetos que afectan a la jugabilidad de forma activa –** Modelos 3d con los que el jugador debe interactuar directamente, mediante los mecanismos de jugabilidad del mini-juego, para superarlo. En el mini-juego de "Recoger los juguetes" son los

propios juguetes a recoger y el baúl de los juguetes donde meterlos. En el minijuego de “Barrer el suelo” son la escoba con la que se barre, las pelusas y el rincón donde hay que depositarlas. En el mini-juego de “Limpiar los cristales” son el trapo que se utiliza para limpiar las manchas y las propias manchas.

El objetivo de este proceso de diseño es aplicar la dirección artística planteada en apartados anteriores de tal forma que los objetos y escenarios diseñados sean aptos para cumplir todas las competencias anteriormente comentadas.

El proceso de diseño se refleja de forma desglosada, mostrando sus partes más relevantes referentes a la apariencia gráfica y animaciones que puedan tener:

Apariencia de los objetos y escenarios

El diseño de la apariencia gráfica de los objetos y escenarios está sujeto a la dirección artística especificada en la fase de concepción (modelos simples, *low poly*, técnicas *cell shading*, etc.), a su condición dentro de los grupos comentados anteriormente y, en menor medida, a posteriores decisiones que se puedan llegar a tomar en la fase de producción. Esto es debido a que como algunos de los objetos considerados afectan directa o indirectamente a la jugabilidad, es necesario considerar otros aspectos de la misma en la producción, ya sea al realizar pruebas internas durante la implementación de las fases de cada mini-juego o pruebas externas con el prototipo de software una vez acabado. Por este motivo la apariencia de estos objetos y escenarios (en concreto de los que afectan a la jugabilidad) ha de tomarse como una referencia, más artística que funcional, de cara a la producción; con posibilidad de cambios posteriores de algunas de sus características, como el tamaño, color, posicionamiento dentro del escenario, etc.

Para llevar a cabo este diseño se realizan bocetos de arte conceptual de los distintos aspectos de los posibles objetos y escenarios propuestos. A continuación se muestran algunos de los más relevantes realizados durante esta fase de diseño:

- **Escenarios.** Muestra de cada escenario propuesto con perspectiva pseudo isométrica. De izquierda a derecha y de arriba abajo: La habitación de Jaimito para el mini-juego “Recoger los juguetes”, el salón de la casa de Jaimito para el mini-juego “Barrer el suelo” y el establo/granero de la granja de Jaimito para el mini-juego “Limpiar los cristales”.

Figura 3.5: Muestra de arte conceptual de los escenarios de CleanQuest

- **Objetos decorativos.** Muestra de algunos bocetos de los objetos decorativos.

Figura 3.6: Muestra de arte conceptual de los objetos decorativos de CleanQuest

- **Objetos que afectan a la jugabilidad de forma pasiva.** Muestra de algunos bocetos de los objetos decorativos.

Figura 3.7: Muestra de arte conceptual de los objetos que afectan a la jugabilidad pasivamente de CleanQuest

- **Objetos que afectan a la jugabilidad de forma activa.** De izquierda a derecha y de arriba abajo: Muestra de algunos juguetes y el baúl del mini-juego “Recoger los juguetes”; muestra de algunas unidades de suciedad (pelusas) y de la escoba del mini-juego “Barrer el suelo”; y muestra de algunas manchas de barro y el trapo del mini-juego “Limpiar los cristales”. El diseño de estos objetos en particular es especial, ya que son los objetos con los que el jugador debe interactuar directamente a través de los mecanismos de la jugabilidad para lograr su objetivo en el mini-juego. Por este motivo, éstos son diseñados de forma que destaquen visualmente, pudiéndolos diferenciar fácilmente del escenario u otros objetos que contenga e identificar su propósito dentro del mini-juego sin problemas. Para ello se utilizan diversos trucos o técnicas en su diseño, como por ejemplo rodear los juguetes o las pelusas en un trazo/halo de un color vivo en vez de negro (aprovechando el estilo *cell shading*).

Figura 3.8: Muestra de arte conceptual de los objetos que afectan a la jugabilidad activamente de CleanQuest

Animaciones de los objetos y escenarios

Como se comentó anteriormente, al considerar las animaciones para todos los elementos (modelos 3d e interfaces) del juego, éstas se pueden clasificar en dos grupos: Animaciones implementadas de forma programática mediante *Unity* y animaciones ligadas al modelo, realizadas con un programa externo. Las animaciones de los objetos y escenarios, debido a su diversidad, pertenecen a ambos grupos. Al igual que el diseño de la apariencia visual de los objetos y escenarios, el diseño definitivo de las animaciones está sujeto a la posterior implementación y pruebas del propio juego en la fase producción. Esto es debido a que ciertos objetos y escenarios afectan a la jugabilidad, por lo que estos diseños deben tomarse como una referencia base, más artística que funcional, de cara a la producción.

Como muchos de estos objetos afectan a la jugabilidad, la mayoría de sus animaciones deben ser implementadas de forma programática con *Unity*. Para el diseño de sus animaciones, con el fin de facilitar el desarrollo respecto a los recursos y experiencia iniciales, se siguen los siguientes principios:

- Reducir al mínimo el número de animaciones necesarias, considerando sólo los objetos esenciales que deben ser animados.
- Simplificar cada una de las animaciones lo más posible.
- Construir animaciones más complejas dentro de *Unity* de forma programática, combinando animaciones simples, reproduciéndolas con bucles o aprovechando sus capacidades de simulación física procedural.

Igualmente todavía se tienen en cuenta los principios de diseño establecidos para las animaciones de los personajes no jugables, en lo que respecta a las animaciones ligadas al modelo realizadas con un programa externo.

3.2.3 Diseño de sonido

El sonido y la música en *CleanQuest* contribuyen, por un lado, a reforzar el atractivo del juego (su ambientación y trasfondo de cara al jugador) y por otro, de manera más funcional, a transmitir al jugador aspectos de las funcionalidades o características del juego. Para su diseño, debido a los recursos y experiencia limitados sobre todo en este aspecto (no se posee ninguna experiencia de composición de música y se carece de los recursos y la habilidad necesarios para realizar grabaciones de voz en condiciones), se seguirán los siguientes principios:

- Reducir al mínimo el número de sonidos y pistas de música necesarias, reutilizándolos cuando sea conveniente.
- Escoger las pistas de música y sonido de forma externa, dando prioridad a los repositorios gratuitos de internet.
- Simplificar las grabaciones de sonido lo más posible (duración e interpretación) y “trocearlas” para tener más control sobre ellas durante la grabación y al gestionarlas en *Unity*.

3.2.4 Diseño del guión

Aunque escasas, *CleanQuest* presenta ciertas situaciones en las que es necesario un guión diseñado de forma adecuada. Estas situaciones son:

- Conversación de cada uno de los familiares de Jaimito con Jaimito al inicio de cada fase en cada mini-juego. En ella se explica el objetivo del mini-juego de forma sutil (sin especificar explícitamente los controles), intercalando fragmentos del trasfondo.
- Conversación de cada uno de los familiares de Jaimito con Jaimito al final de cada fase en cada mini-juego. En ella felicita a Jaimito (el jugador) por haber completado la fase y le presenta los premios obtenidos.
- Conversación de cada uno de los familiares de Jaimito con Jaimito al finalizar las 3 fases de un mini-juego. En ella el familiar de Jaimito que asiste en el mini-juego, se comunica con el jugador, le explica en qué consiste el desafío de realidad aumentada referente a la tarea que representa ese mini-juego y le invita a participar en él.

- Explicación dirigida a los padres o tutores del jugador, en el control parental, sobre su sentido dentro del juego, el papel de los cromos y su relación con el desafío de realidad aumentada.

En estos casos el guión es importante, ya que se encarga de reforzar el trasfondo, la ambientación y por consiguiente el atractivo del juego. Además es el medio por el cual se explica directamente al jugador aspectos de la funcionalidad (características o jugabilidad) del juego. Igualmente su diseño no es trivial, ya que su contenido está sujeto a la caracterización de cada personaje no jugable, a su entorno (trasfondo) y a la vez a la funcionalidad útil que se quiere transmitir. En otras palabras, no es sencillo diseñar un guión mediante el cual se informe al jugador de los objetivos o funcionalidades útiles del juego de forma lo suficientemente sutil para no sacar a los familiares de Jaimito "de su personaje".

3.2.5 Diseño de la interfaz de usuario

En la industria del videojuego apenas existen estándares para el diseño de interfaces de usuario, a diferencia de los productos de software convencional. Aunque se suele partir de unas pautas básicas universales de diseño (tamaño de fuente óptimo, contraste de colores, visibilidad, etc.), el diseño de la interfaz es personal de cada videojuego y refleja un aspecto de la dirección artística empleada en el mismo.

En *CleanQuest,* el diseño de la interfaz de usuario propuesto está sujeto al hardware elegido (dispositivos móviles con pantalla táctil), al tipo de usuario objetivo (niños de 6 a 9 años) y a la dirección artística en general. El proceso de diseño de la interfaz de usuario se refleja de forma desglosada, mostrando sus partes más relevantes:

Apariencia general

- **Fuente.** Se proponen 2 tipos de fuentes, una general, empleada en la mayoría de los textos del juego y una puntual empleada en ciertos textos. La general, de nombre *DNK*, descargada de un repositorio gratuito, ha sido elegida por su aspecto simple, voluminoso y desenfadado; ya que encaja perfectamente con el resto de los elementos del juego (sobre todo al considerar las técnicas *cell shading* utilizadas). La fuente puntual, de nombre Arial, ha sido elegida por su aspecto simple y neutro para ser utilizada en casos puntuales en los que el usuario no deba tener ningún problema en entender el texto. Por ejemplo en la introducción de códigos por parte de los padres o tutores del jugador en el control parental, para el desafío de realidad aumentada. El tamaño de ambas fuente no es siempre el mismo, ya que será adaptado según el elemento gráfico que la utilice (botones, subtítulos, cuadros de texto, etc.).

- **Botones virtuales táctiles.** Se propone un diseño general para los botones. Está pensado para ser llamativo y su contenido lo suficientemente intuitivo para que el jugador objetivo pueda reconocer el propósito del botón al instante. Para ello se siguen los siguientes principios:

- Se utilizan imágenes o símbolos en vez de texto para el contenido del botón siempre que sea posible.
- Se utiliza un tamaño y forma de botón variable, pero siempre lo suficientemente grande respecto a los otros elementos de la interfaz para que pueda ser reconocido por el jugador objetivo en todo tipo de dispositivos móviles y, si el botón contiene texto, que pueda ser leído sin dificultad.

- El estilo visual del botón está basado en el diseño del objeto "Baúl de los juguetes" (figura 3.8), tonos azules y amarillos intensos y aspecto de plástico o textura barnizada. Se utilizará una variante de este estilo, sustituyendo el tono amarillo por uno más anaranjado para botones especiales, que denoten la finalidad de una sucesión (Ej. salir de un mini-juego, o último botón de pasar página) o que activen funcionalidades externas o particulares en el juego (como una entrada de texto).

Figura 3.9: Muestra del diseño de botones virtuales táctiles de CleanQuest

- **Fondos y ventanas o pantallas secundarias.** Se propone un diseño general para los fondos de las pantallas principales y para las sub-pantallas o pantallas secundarias. En el caso de los fondos se utilizará una trama genérica de tablones de madera, que recuerda a la típicamente utilizada en las edificaciones de madera en una granja. En el caso de las sub-pantallas o pantallas secundarias se utilizarán fragmentos de papel cuadriculado, como si fueran páginas arrancadas de una libreta o cuaderno escolar infantil.

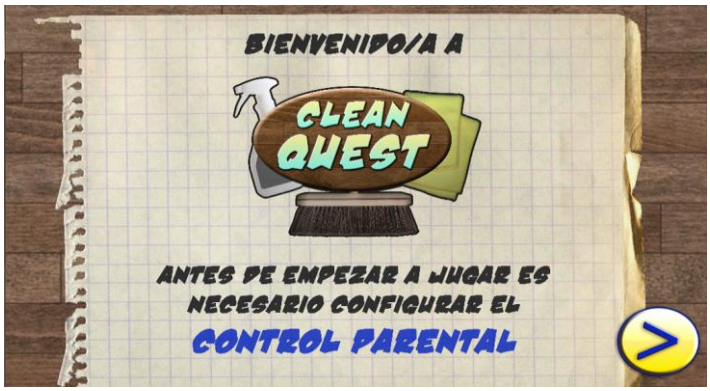


Figura 3.10: Mockup de una pantalla de CleanQuest

Apariencia y animaciones por pantallas

La apariencia y animaciones de la interfaz de usuario para cada pantalla individual de *CleanQuest* se especificarán más adelante, en el apartado de diseño de implementación y técnico (apartado 3.4). En él se construirá un prototipo del videojuego de bajo nivel de fidelidad a base de *mockups* de estas pantallas. Mediante

este prototipo se presentará el aspecto visual de cada pantalla y su justificación de cara a su propósito.

3.2.6 Diseño del control parental, cromos coleccionables y dinámica de premios

Como se comentó anteriormente, en *CleanQuest* el jugador puede obtener premios en forma de cromos virtuales coleccionables completando las fases de los mini-juegos o superando el "Desafío de realidad aumentada" de cualquier mini-juego en la vida real, al introducir los padres o tutores del jugador el código/contraseña respectivo que introdujo durante el control parental.

Este aspecto del juego (la relación entre el control parental, los cormos coleccionables y la dinámica de premios) no es trivial y necesita un apartado dedicado para comentar su diseño.

Diseño del Control parental

Las pantallas del control parental aparecen únicamente al iniciar por primera vez el juego. Su cometido es, primeramente, llamar la atención de los padres, esto es, si es su hijo/a quien inicia el juego por primera vez, avisarle que deje el dispositivo móvil a sus padres o tutores. A continuación, debe informar a los padres o tutores sobre las características principales del juego, el sentido de los cromos virtuales coleccionables y del "Desafío de realidad aumentada" de cara a resolver el problema propuesto, para que todo pueda funcionar. Por último, permitirá a los padres o tutores establecer los códigos/contraseñas para cada tarea del "Desafío de realidad aumentada" y, opcionalmente, el lugar del entorno doméstico donde desean que su hijo/a las realice.

Teniendo esto en cuenta y atendiendo al diseño de la interfaz de usuario general, se propone la siguiente estructura (pantallas), flujo y aspecto visual para el control parental. Consta de una sucesión de 3 pantallas principales a modo de páginas de un libro o cuaderno. El usuario podrá pasar a la página siguiente o a la anterior mediante botones en los laterales. El contenido y propósito de cada una de las pantallas es el siguiente:

- **1ª Pantalla – Bienvenida.** Pantalla que da la bienvenida al usuario que inicia por primera vez el juego (figura 3.10).

- **2ª Pantalla – Presentación del control parental.** Pantalla que reproduce un video introductorio compuesto de imágenes, animaciones y una voz en off neutra como narrador. En este video, primeramente se avisa de que el control parental es competencia de los padres o tutores del jugador y, en el caso de que sea jugador el que tiene el dispositivo móvil en ese momento, que se lo deje a sus padres o tutores. A continuación el video introduce a los padres o tutores sobre las características principales del juego; el sentido de los cromos virtuales coleccionables y del "Desafío de realidad aumentada" de cara a resolver el problema propuesto; y la necesidad de que ellos introduzcan los códigos/contraseñas y los lugares físicos respectivos para que todo pueda

funcionar. Se permite al usuario volver a reproducir este video con un botón en el centro de la pantalla.

Figura 3.11: Mockup de la pantalla de presentación del control parental en CleanQuest

El motivo de utilizar un video o *cutscene* introductorio en vez de texto (o texto acompañado de imágenes) para explicar todo el funcionamiento, como ya lo hicieron las plataformas *Choremonster* o *Family Chores*, es debido a que se trata de un videojuego enfocado a un público casual (incluyendo los padres o tutores). Este tipo de usuarios, por lo general, no están acostumbrados de leer grandes cantidades de texto y, en este caso, toda su intención y concentración está puesta para iniciar el juego lo más rápido posible. Por lo que un vídeo, sucesión de imágenes con audio narrativo, se considera la mejor opción para hacer llegar al usuario objetivo toda esta carga de información.

- **3ª Pantalla – Introducción de códigos y lugares.** Última pantalla del control parental. En ella se permite a los padres o tutores del jugador introducir los códigos/contraseñas y los lugares para las 3 tareas del "Desafío de realidad aumentada". Para ello se muestran tres botones táctiles que representan cada una de las tareas del desafío de realidad aumentada (que a su vez representan a los 3 mini-juegos). Al pulsar uno aparecerá, mediante una animación, una pantalla secundaria con un *prompt* que permitirá introducir el código/contraseña y el lugar físico designado para el "Desafío de realidad aumentada" de esa tarea en particular. Una vez que se hayan introducido correctamente los códigos/contraseñas y lugares de las 3 tareas (se denotará mediante un halo verde alrededor de cada botón) se desbloqueará el botón de paso al menú principal, terminando de esta forma el control parental.

Figura 3.12: Mockup de la pantalla de introducción de códigos/contraseñas y lugares del control parental

Más adelante, una vez que el jugador complete una de las tareas del "Desafío de realidad aumentada" en la vida real (en el lugar establecido para esa tarea), y sus padres o tutores lo confirmen, los mismos podrán introducir el código/contraseña correspondiente a esa tarea en una sección de la galería de cromos para desbloquear cromos coleccionables de mayor valor.

Figura 3.13: Mockup de la sección de introducción de códigos en la galería de cromos

Diseño de los cromos virtuales coleccionables

Los **cromos coleccionables virtuales**, como se comentó anteriormente, son el incentivo que hace de nexo entre los aspectos de tipo tablas de tareas y de simulación de *CleanQuest*. Éstos se ofrecen como premios al completar las fases de un mini-juego o al completar alguna tarea del "Desafío de realidad aumentada". Su contenido está basado en los personajes u objetos que pueblan toda la ambientación y el trasfondo del juego: juguetes, animales, familiares de Jaimito, utensilios de la granja, etc. El estilo visual del cromo es sencillo, sigue un diseño similar al de la interfaz de usuario, para encajar con el resto de los elementos y la ambientación general del juego. Un cromo virtual coleccionable en *CleanQuest* presenta los siguientes elementos:

Figura 3.14: Ejemplo de cromo virtual coleccionable en CleanQuest

- **(1)** Marco con estilo de tablones de madera.
- **(2)** Ilustración en 3d de su contenido.

- **(3)** Número del cromo respecto a la colección completa.
- **(4)** Rareza del cromo (Especial, Oro, Plata, Bronce o Juguete; su sentido se comentará más adelante en el diseño de la dinámica de premios).

La colección de cromos virtuales que tenga un jugador se puede ver en detalle en la sección de **"Galería de cromos"**. En esta galería, aparte de poder introducir los códigos/contraseñas de las tareas del "Desafío de realidad aumentada" aumentada para desbloquear cromos nuevos, se pueden ver todos los cromos coleccionados, ordenados como si de un álbum se tratase. Al pulsar sobre cada uno de ellos aparecerá en el lateral derecho de la pantalla el modelo 3d correspondiente al contenido del cromo (a modo de visor de modelos 3d) y su nombre en el lateral inferior.

Figura 3.15: Mockup de la galería de cromos coleccionables de CleanQuest

Diseño de la dinámica de premios

Como se comentó en la fase de concepción (apartado 3.1), en lo que respecta a la jugabilidad y reglas de *CleanQuest*, para no provocar el rechazo del jugador objetivo (considerando su rango de edad) debido a la penalización del contador progresivo de tiempo, se ha optado por premiar siempre al jugador. Se le premia independientemente de la habilidad que haya demostrado en la fase del mini-juego, aunque para ello tenga que recibir un premio de menor valor. En este caso, sería absurdo considerar como premios directamente los cromos virtuales coleccionables, ya que una vez obtenido uno, no tendría sentido acumular otro igual de cara a la galería. Por este motivo **se ha decidido utilizar premios simbólicos, en forma de trofeos temporales** para cada partida, que se otorgarán al jugador además de los cromos. Estos trofeos tienen un aspecto visual típico de una copa de metal, fácilmente reconocibles como tales.

Figura 3.16: Aspecto visual de un trofeo en CleanQuest

Como ya se mencionó en el diseño de los cromos, tanto los cromos coleccionables como los trofeos de *CleanQuest,* tendrán asignado un **valor o rareza**. Este valor representará la habilidad demostrada por el jugador al completar la fase de un mini-juego. Según las reglas comunes de todos los mini-juegos establecidas en la fase de concepción (apartado 3.1), cuanto menos tarde el jugador en completar una fase ganará un premio de más valor, pero siempre optará a uno. Cuando un jugador obtiene un trofeo de una rareza, obtendrá, si no los tiene ya, el cromo coleccionable respectivo de esa fase de igual rareza y todos cromos restantes de esa fase de rareza inferior. A continuación se muestra una tabla que describe todos los posibles valores y rarezas de los cromos coleccionables y los trofeos en *CleanQuest,* de mayor a menor nivel de rareza:

Rareza	Trofeo	Cromo	Obtención
Especial	-		*Obtenido al superar alguno de los desafíos de realidad aumentada*
Oro			*Obtenido al alcanzar el mejor tiempo en una fase de un mini-juego.*
Plata			*Obtenido al alcanzar el segundo mejor tiempo en una fase de un mini-juego o un trofeo superior.*
Bronce			*Obtenido al alcanzar el tercer mejor tiempo en una fase de un mini-juego o un trofeo superior*
Juguete			*Obtenido por defecto en una fase de un mini-juego si no se consigue otro trofeo o consiguiendo un trofeo superior*

Tabla 3.1: Niveles de rarezas en los trofeos y cromos coleccionables de CleanQuest

3.3 Preproducción – Especificación de requisitos

Una vez que se han tomado todas las decisiones de diseño iniciales, esto es, las de las fases de concepción y de diseño conceptual y artístico, es posible realizar una especificación de requisitos. El objetivo de estos requisitos es ser utilizados como base o guía para la fase de diseño de más peso de cara al desarrollo, el diseño de implementación y técnico (apartado 3.4), y para la posterior fase pruebas (apartado 4) de la producción, permitiendo a partir de los mismos idear una serie de pruebas adecuadas para validar el prototipo de software de alto nivel de fidelidad.

Para la formalización y extracción de estos requisitos se utiliza como referencia el documento de **Métrica v3** (administración electrónica del gobierno [47]). Cabe destacar que aunque este documento está orientado a una metodología más pesada (en contraposición con la combinación de metodologías principalmente ágiles empleadas en *CleanQuest*) se ha decidido utilizarlo debido a que, por un lado, es el documento de especificación de requisitos del que se posee más experiencia inicial y por otro, es fácilmente compatible con cualquier otro tipo de metodologías, **mientras se consideren requisitos con un grado de abstracción suficiente para garantizar su estabilidad respecto al producto en general**.

En este apartado se recogen tanto los requisitos funcionales, orientados a la implementación de las funcionalidades del juego, como los requisitos no funcionales, orientados a establecer propiedades o atributos del mismo de forma definitiva. Se utilizará un modelo de requisito con la siguiente estructura o formato:

Identificador	Identificador del requisito	**Nombre**	Nombre del requisito
Prioridad	Alta/Media/Baja	**Repercusión**	Alta/Media/Baja
Dependencias	Identificadores de los requisitos con los que guarda dependencias.	**Fecha**	dd/mm/aaaa
Descripción		Descripción breve y concisa del requisito, orientada para su posterior implementación.	

Figura 3.17: Formato de requisito propuesto

- **Identificador.** Código que identifica unívocamente al requisito respecto a los demás. El formato de este código es el siguiente: R(Ti)[Sub-nf]-XXX; (Ti) puede ser “F” para los requisitos funcionales o “NF” para los no funcionales. Dentro de los no funcionales, [Sub-nf] representa el sub-tipo de requisito, en este caso puede ser “D” para los de diseño, “I” para los de interfaz o “R” para los de rendimiento. XXX es el número del requisito, de 000 a 999, incremental, ya que no se estiman más de 1000 requisitos en total. Ejemplos de estos identificadores podrían ser “RF-003”, “RNFD-120” o “RNFR-335”.

- **Nombre.** Nombre narrativo basado en la descripción que identifica al requisito.
- **Prioridad.** Prioridad o urgencia que presenta el requisito a la hora de implementarlo en la fase de producción. Sus posibles valores son Alta, Media o Baja.
- **Repercusión.** Dependencia que tiene el requisito respecto al desarrollo en su totalidad. En otras palabras, el coste teórico que acarrearía para el desarrollo una modificación del requisito. Sus posibles valores son Alta, Media o Baja.
- **Dependencias.** Dependencia o relación que tiene el requisito con otro u otros requisitos. Se incluye los identificadores de los requisitos con los que guarde dependencia.
- **Fecha.** Fecha de redacción del requisito.
- **Descripción.** Descripción breve y concisa del requisito, orientada para su posterior implementación.

3.3.1 Requisitos funcionales

Identificador	RF-001	**Nombre**	*Inicio de la aplicación - general*
Prioridad	Alta	**Repercusión**	Baja
Dependencias	-	**Fecha**	29/09/14
Descripción	La primera vez que se inicie la aplicación debe aparecer la pantalla del control parental. Una vez completado éste por el usuario, las subsiguientes veces que se inicie la aplicación deberá iniciarse en la pantalla del menú principal.		

Identificador	RF-002	**Nombre**	*Navegación - control parental*
Prioridad	Alta	**Repercusión**	Baja
Dependencias	RNFR-001	**Fecha**	30/09/14
Descripción	La aplicación permitirá al usuario navegar sin problemas por las 3 pantallas del control parental en ambos sentidos, mediante botones.		

Identificador	RF-003	**Nombre**	*Video introductorio - control parental*
Prioridad	Alta	**Repercusión**	Baja
Dependencias	RF-002	**Fecha**	30/09/14
Descripción	Cuando el usuario llegue por primera vez a la 2ª pantalla del control parental, deberá aparecer el video introductorio de *CleanQuest*. Al terminar se deberá dar la opción de volver a visionarlo mediante un botón en esa pantalla.		

Identificador	RF-004	**Nombre**	*Introducción códigos - control parental*
Prioridad	Alta	**Repercusión**	Alta
Dependencias	RF-002	**Fecha**	30/09/14
Descripción		En la 3ª pantalla del control parental, la aplicación deberá permitir al usuario introducir los códigos y los lugares físicos para cada tarea del desafío de realidad aumentada, mediante un *prompt*, al pulsar el botón correspondiente que representa a esa tarea.	

Identificador	RF-005	**Nombre**	*Reconocimiento códigos - control parental*
Prioridad	Media	**Repercusión**	Media
Dependencias	RF-001, RF-002 y RF-004	**Fecha**	30/09/14
Descripción		En la 3ª pantalla del control parental, la aplicación deberá reconocer si el usuario ha introducido los códigos y los lugares físicos para cada tarea del desafío de realidad aumentada correctamente (y guardarlos para un posterior uso), mostrándolo con un halo de color verde alrededor del botón de esa tarea o rojo en el caso contrario.	

Identificador	RF-006	**Nombre**	*Barrera códigos/contraseñas - control parental*
Prioridad	Media	**Repercusión**	Baja
Dependencias	RF-001, RF-002, RF-004 y RF-005	**Fecha**	30/09/14
Descripción		En la 3ª pantalla del control parental, la aplicación deberá bloquear el paso hacia el menú principal hasta que no se hayan introducido correctamente todos los códigos y lugares de las 3 tareas. Una vez hecho deberá permitir acceder al menú principal	

Identificador	RF-007	**Nombre**	*Navegación – menú principal*
Prioridad	Alta	**Repercusión**	Baja
Dependencias	RF-001 y RNFR-001	**Fecha**	01/10/14
Descripción		En el menú principal, la aplicación debe permitir al usuario el acceso a la galería de cromos y a los tres mini-juegos de mediante botones. Igualmente debe permitir salir de la propia aplicación mediante otro botón.	

Identificador	RF-008	Nombre	*Sub-pantalla de fases – menú principal*
Prioridad	Alta	**Repercusión**	Media
Dependencias	RF-007	**Fecha**	01/10/14
Descripción	En el menú principal, cuando el usuario pulse uno de los botones de acceso a los mini-juegos, la aplicación deberá mostrar una sub-pantalla con botones de acceso a las fases que hayan sido desbloqueadas de ese mini-juego. Cada uno de estos botones tendrá situado a su lado un icono que muestre el trofeo de más valor alcanzado en esa fase.		

Identificador	RF-009	Nombre	*Navegación – galería de cromos*
Prioridad	Media	**Repercusión**	Baja
Dependencias	RF-007 y RNFR-001	**Fecha**	02/10/14
Descripción	En la galería de cromos, la aplicación debe permitir al usuario el acceso a las sub-pantallas de visor de cromos y de introducción de códigos del desafío de realidad aumentada, mediante botones. Igualmente debe permitir volver al menú principal mediante otro botón.		

Identificador	RF-010	Nombre	*Visor de cromos – galería de cromos*
Prioridad	Alta	**Repercusión**	Alta
Dependencias	RF-009	**Fecha**	02/10/14
Descripción	En la sub-pantalla de visor de cromos de la galería de cromos, la aplicación debe mostrar todos los cromos coleccionables obtenidos por el usuario. Al pulsar en uno de ellos, la aplicación deberá mostrar el modelo 3d que representa el contenido del cromo. Igualmente debe permitir salir de la sub-pantalla mediante otro botón.		

Identificador	RF-011	Nombre	*Introducción de códigos – galería de cromos*
Prioridad	Alta	**Repercusión**	Baja
Dependencias	RF-009	**Fecha**	02/10/14
Descripción	En la sub-pantalla de introducción de códigos de la galería de cromos, la aplicación debe permitir al usuario introducir los códigos de cada desafío de realidad aumentada por medio de un único botón en la página. Igualmente debe permitir salir de la sub-pantalla mediante otro botón.		

Identificador	RF-012	**Nombre**	*Obtención de cromos – galería de cromos*
Prioridad	Media	**Repercusión**	Media
Dependencias	RF-005, RF-009, RF-010, RF-011 y RF-013	**Fecha**	02/10/14
Descripción		En la sub-pantalla de introducción de códigos de la galería de cromos, al introducir un código para un cromo, la aplicación debe reconocerlo y mostrar al usuario el cromo obtenido si el código es correcto (coincide con alguno) o un mensaje de error si es incorrecto.	

Identificador	RF-013	**Nombre**	*Obtención de cromos – general*
Prioridad	Alta	**Repercusión**	Alta
Dependencias	RF-012 y RF-017	**Fecha**	02/10/14
Descripción		Cada vez que el usuario desbloquee un cromo coleccionable, ya sea como premio en los mini-juegos o introduciendo un código en la galería de cromos, la aplicación debe guardar una referencia de este cromo para su posterior uso.	

Identificador	RF-014	**Nombre**	*Desbloqueo de fases – menú principal*
Prioridad	Media	**Repercusión**	Baja
Dependencias	RF-008	**Fecha**	02/10/14
Descripción		La aplicación debe reconocer si un usuario ha completado una fase de un mini-juego (a partir de la primera) al menos una vez, antes de permitir el acceso a ella directamente desde la sub-pantalla de fases del menú principal.	

Identificador	RF-015	**Nombre**	*Inicio de fase – general mini-juegos*
Prioridad	Media	**Repercusión**	Baja
Dependencias	RF-008	**Fecha**	02/10/14
Descripción		Al iniciar una fase de cualquier mini-juego, la aplicación debe reproducir la *cutscene* de inicio de fase correspondiente. Esta *cutscene* se reproducirá indefinidamente hasta que el jugador decida empezar la fase jugable mediante un botón.	

Identificador	RF-016	**Nombre**	*Salir de fase – general mini-juegos*
Prioridad	Alta	**Repercusión**	Baja
Dependencias	-	**Fecha**	02/10/14
Descripción		Dentro de una fase de un mini-juego o en la *cutscene* de desafío de realidad aumentada, la aplicación debe permitir al usuario volver al menú principal en cualquier momento, mediante un botón.	

Identificador	RF-017	**Nombre**	*Final de fase – general mini-juegos*
Prioridad	Media	**Repercusión**	Alta
Dependencias	RF-013 y RF-014	**Fecha**	02/10/14
Descripción	Al terminar una fase de un mini-juego, la aplicación debe reproducir la *cutscene* de final de fase correspondiente. En ella deberá mostrar al usuario el trofeo y el/los cromos obtenidos como premios en la fase. Esta *cutscene* se reproducirá indefinidamente hasta que el jugador decida empezar la siguiente fase jugable mediante un botón.		

Identificador	RF-018	**Nombre**	*Control tiempo y premios – general mini-juegos*
Prioridad	Alta	**Repercusión**	Alta
Dependencias	RF-017	**Fecha**	02/10/14
Descripción	Dentro de una fase de un mini-juego, la aplicación deberá registrar el paso del tiempo en segundos, mostrando un contador progresivo en la interfaz de usuario. Igualmente la aplicación deberá registrar y mostrar un icono al lado del contador de tiempo del trofeo actual y del trofeo siguiente al que opta el usuario en cada rango de tiempo establecido para esa fase en particular.		

Identificador	RF-019	**Nombre**	*Reconocimiento de premios – general mini-juegos*
Prioridad	Alta	**Repercusión**	Alta
Dependencias	RF-017 y RF-018	**Fecha**	02/10/14
Descripción	Al terminar una fase de un mini-juego, antes de la cutscene de final de fase, la aplicación deberá reconocer el último trofeo al que optaba el usuario y los cromos coleccionables a los que opta con ese trofeo. Comprobará si alguno de esos cromos es repetido (ya ha sido conseguido anteriormente), para descartarlo. Deberá registrar estos premios hasta la *cutscene* de final de fase.		

Identificador	RF-020	**Nombre**	*Control de puntuación – general mini-juegos*
Prioridad	Alta	**Repercusión**	Alta
Dependencias	RF-017, RF-018 y RF-021	**Fecha**	02/10/14
Descripción	Dentro de una fase de un mini-juego, la aplicación deberá registrar el número actual de hitos o acciones numerables que va alcanzando el jugador. Este número deberá ser mostrado en la interfaz de usuario en tiempo real, acompañado del número total necesario de hitos para esa fase.		

Identificador	RF-021	**Nombre**	*Reconocimiento final de fase – general mini-juegos*
Prioridad	Alta	**Repercusión**	Alta
Dependencias	RF-017, RF-018, RF-019 y RF-020	**Fecha**	02/10/14
Descripción	Dentro de una fase de un mini-juego, una vez que el número de hitos o acciones que va realizando el jugador alcance el número de hitos o acciones total establecido para esa fase, la fase deberá terminar. En otras palabras, la aplicación deberá parar la jugabilidad, realizar todos los preparativos necesarios y pasar automáticamente a la *cutscene* de final de fase.		

Identificador	RF-022	**Nombre**	*Desafío de realidad aumentada – general mini-juegos*
Prioridad	Baja	**Repercusión**	Baja
Dependencias	RF-017	**Fecha**	02/10/14
Descripción	Al iniciar finalizar las tres fases de un mini-juego, la aplicación debe reproducir la *cutscene* de presentación del desafío de realidad aumentada para ese mini-juego. Esta *cutscene* se reproducirá indefinidamente hasta que el jugador decida volver al menú principal mediante un botón.		

Identificador	RF-023	**Nombre**	*Gestión de assets – general mini-juegos*
Prioridad	Alta	**Repercusión**	Alta
Dependencias	RF-018, RF-020, RNFD-004 y RNFI-008	**Fecha**	02/10/14
Descripción	La aplicación deberá generar todos los *assets* (cámara, controlador de jugador, *rigid bodys*, etc.) en cada fase de un mini-juego y actualizarlos de forma correcta cada iteración o *frame* de la ejecución de acuerdo a las especificaciones (posición, tamaño, etc.) de ese mini-juego.		

Identificador	RF-024	**Nombre**	*Control de usuario – jugabilidad "Recoger los juguetes"*
Prioridad	Alta	**Repercusión**	Alta
Dependencias	RF-023 y RNFR-001	**Fecha**	03/10/14
Descripción	La aplicación deberá permitir al usuario controlar los juguetes del mini-juego "Recoger los juguetes", agarrándolos, sujetándolos y soltándolos mediante mecanismos *drag and drop,* de forma físicamente coherente, hasta depositarlos dentro del baúl de los juguetes.		

Identificador	RF-025	**Nombre**	*Reconocimiento de hitos – jugabilidad "Recoger los juguetes"*
Prioridad	Alta	**Repercusión**	Media
Dependencias	RF-20, RF-023, RF-024 y RNFR-001	**Fecha**	03/10/14
Descripción		En las fases del mini-juego "Recoger los juguetes", cuando el jugador deposite un juguete dentro del baúl de los juguetes, la aplicación deberá reconocerlo como hito o acción numerable de cara a la puntuación, incrementándola en 1.	

Identificador	RF-026	**Nombre**	*Control de usuario – jugabilidad "Barrer el suelo"*
Prioridad	Alta	**Repercusión**	Alta
Dependencias	RF-023 y RNFR-001	**Fecha**	03/10/14
Descripción		La aplicación deberá permitir al usuario controlar la escoba del mini-juego "Barrer el suelo", mediante un, *joystick* virtual táctil y dos botones de control, conduciéndola en cualquier dirección, de forma físicamente coherente.	

Identificador	RF-027	**Nombre**	*Interacción pelusas – jugabilidad "Barrer el suelo"*
Prioridad	Alta	**Repercusión**	Alta
Dependencias	RF-023, RF-026 y RNFR-001	**Fecha**	03/10/14
Descripción		La aplicación deberá permitir al usuario, en el mini-juego "Barrer el suelo", ir apartando las pelusas de forma físicamente coherente con la escoba.	

Identificador	RF-028	**Nombre**	*Reconocimiento de hitos – jugabilidad "Barrer el suelo"*
Prioridad	Alta	**Repercusión**	Media
Dependencias	RF-20, RF-023, RF-026, RF-027 y RNFR-001	**Fecha**	03/10/14
Descripción		En las fases del mini-juego "Barrer el suelo", cuando el jugador conduzca una pelusa hasta el rincón indicado (donde se encuentra el recogedor), la aplicación deberá reconocerlo como hito o acción numerable de cara a la puntuación, incrementándola en 1.	

Identificador	RF-029	**Nombre**	*Control de usuario – jugabilidad "Limpiar los cristales"*
Prioridad	Alta	**Repercusión**	Alta
Dependencias	RF-023 y RNFR-001	**Fecha**	03/10/14
Descripción	La aplicación deberá permitir al usuario controlar el trapo del mini-juego "Limpiar los cristales", pudiendo agarrarlo, soltarlo y restregarlos contra las manchas, mediante mecanismos *drag and drop,* de forma físicamente coherente, hasta que termine de limpiar todas las manchas.		

Identificador	RF-030	**Nombre**	*Desvanecimiento de manchas – jugabilidad "Limpiar los cristales"*
Prioridad	Alta	**Repercusión**	Alta
Dependencias	RF-023, RF-029 y RNFR-001	**Fecha**	03/10/14
Descripción	La aplicación deberá permitir al usuario, en el mini-juego "Limpiar los cristales", reducir de tamaño (un porcentaje determinado) de una mancha cada vez que restriegue el trapo por encima de ella. Cuando el tamaño de la mancha llegue al mínimo debe desvanecerse por completo.		

Identificador	RF-031	**Nombre**	*Reconocimiento de hitos – jugabilidad "Limpiar los cristales"*
Prioridad	Alta	**Repercusión**	Media
Dependencias	RF-20, RF-023, RF-029 y RF-030	**Fecha**	03/10/14
Descripción	En las fases del mini-juego "Limpiar los cristales", cada vez que el jugador desvanezca una mancha del cristal, la aplicación deberá reconocerlo como hito o acción numerable de cara a la puntuación, incrementándola en 1.		

Identificador	RF-032	**Nombre**	*Gestión de simulación física – general mini-juegos*
Prioridad	Alta	**Repercusión**	Alta
Dependencias	RF-023, RF-024, RF-027 y RF-030	**Fecha**	03/10/14
Descripción	La aplicación deberá aportar a los elementos con cuerpos rígidos de los mini-juegos (juguetes, pelusas y el trapo) una dinámica, respecto a su movimiento y su iteración con otros elementos, coherente de acuerdo a las físicas del mundo real.		

Identificador	RF-033	**Nombre**	*Gestión de límites y colisiones – general mini-juegos*
Prioridad	Alta	**Repercusión**	Alta
Dependencias	RF-023, RF-024 al RF-032	**Fecha**	03/10/14
Descripción	La aplicación deberá generar límites y registrar las colisiones para la interacción de los elementos de cuerpo rígido con el resto de los elementos del escenario. Ya sea para establecer barreras físicas dentro del escenario o para reconocer y clasificar interacciones entre 2 elementos.		

Identificador	RF-034	**Nombre**	*Gestión de límites y colisiones – general mini-juegos*
Prioridad	Alta	**Repercusión**	Alta
Dependencias	RF-023, RF-024 al RF-032	**Fecha**	03/10/14
Descripción	La aplicación deberá generar límites y registrar las colisiones para la interacción de los elementos de cuerpo rígido con el resto de los elementos del escenario. Ya sea para establecer barreras físicas dentro del escenario o para reconocer y clasificar interacciones entre 2 elementos.		

Identificador	RF-033	**Nombre**	*Gestión de límites y colisiones – general mini-juegos*
Prioridad	Alta	**Repercusión**	Alta
Dependencias	RF-023, RF-024 al RF-032	**Fecha**	03/10/14
Descripción	La aplicación deberá generar límites y registrar las colisiones para la interacción de los elementos de cuerpo rígido con el resto de los elementos del escenario. Ya sea para establecer barreras físicas dentro del escenario o para reconocer y clasificar interacciones entre 2 elementos.		

Identificador	RF-034	**Nombre**	*Reproducción de animaciones – general*
Prioridad	Baja	**Repercusión**	Alta
Dependencias	RF-004, RF-005, RF-007, RF-008, RF-012, RF-014, RF-015, RF-017, RF-022, RF-022, RF-023 y RF-027	**Fecha**	04/10/14
Descripción	La aplicación deberá mostrar correctamente las animaciones de todos los elementos que las contengan. Ya sean de la interfaz (aparición de sub-pantallas o *prompts*) o de *assets* dentro de un escenario (PNJs u objetos).		

Identificador	RF-035	Nombre	*Reproducción de sonidos – general*
Prioridad	Baja	**Repercusión**	Baja
Dependencias	RF-023	**Fecha**	04/10/14
Descripción		La aplicación deberá reproducir correctamente las pistas de música, efectos de sonido y clips de audio correspondientes cuando sea necesario.	

Tabla 3.2: Requisitos funcionales de CleanQuest

3.3.2 Requisitos no funcionales

Identificador	RNFR-001	Nombre	*Demora de respuesta - general*
Prioridad	Alta	**Repercusión**	Alta
Dependencias	RF-002, RF-007, RF-009, RF-024 al RF-030	**Fecha**	04/10/14
Descripción		La aplicación deberá responder de forma prácticamente instantánea a las acciones del usuario, ya sea en el control de los mini-juegos o en la navegación entre pantallas de la interfaz.	

Identificador	RNFR-002	Nombre	*Sobrecarga de eventos - general*
Prioridad	Alta	**Repercusión**	Alta
Dependencias	RNFR-001	**Fecha**	04/10/14
Descripción		El hilo principal de cada script que controla la actualización de la escena cada iteración o *frame*, *Update(),* no deberá ser sobrecargado. Se deberán utilizar co-rutinas paralelas cuando sea posible.	

Identificador	RNFR-003	Nombre	*Compatibilidad de formatos - general*
Prioridad	Media	**Repercusión**	Media
Dependencias	RNFR-001 y RNFR-002	**Fecha**	04/10/14
Descripción		La aplicación deberá utilizar formatos de archivos externos (sonidos, imágenes, etc.) que sean compatibles posteriormente con *Android* y que no sean perjudiciales (peso, tolerancia por *Unity*, etc.) de cara al rendimiento.	

Identificador	RNFD-004	Nombre	*Diseño de assets - general*
Prioridad	Alta	Repercusión	Alta
Dependencias	RF-023	Fecha	05/10/14
Descripción	El diseño de todos los *assets* (modelos 3d, animaciones, sonido, interfaz, etc.) deberá respetar las decisiones de diseño tomadas en las fases de concepción y diseño conceptual y artístico.		

Identificador	RFND-005	Nombre	*Android en programación - general*
Prioridad	Alta	Repercusión	Alta
Dependencias	RNFD-004 y RNFR-002	Fecha	05/10/14
Descripción	En la codificación de los scripts, se deberán aplicar las funciones o estructuras exclusivas de *Android* cuando sea necesario, para garantizar su compatibilidad.		

Identificador	RNFD-006	Nombre	*Portabilidad - general*
Prioridad	Alta	Repercusión	Alta
Dependencias	RNFR-003 y RFND-005	Fecha	05/10/14
Descripción	La aplicación deberá poder ser compatible con todas las versiones de Android, a partir de la versión 2.3.1 "*Gingerbread*".		

Identificador	RNFI-007	Nombre	*Resolución interfaz – general*
Prioridad	Alta	Repercusión	Alta
Dependencias	RNFD-004	Fecha	05/10/14
Descripción	La interfaz de usuario deberá poder ser adaptada automáticamente a todas las resoluciones posibles del mercado de dispositivos móviles, con la menor pérdida de calidad posible.		

Identificador	RNFI-008	Nombre	*Diseño de interfaz – general*
Prioridad	Alta	Repercusión	Media
Dependencias	RNFD-004, RNFI-007 y RF-023	Fecha	05/10/14
Descripción	El diseño de la interfaz de usuario, atendiendo a su aspecto visual (legibilidad del texto, contraste de colores, etc.) y funcionalidad (espacio táctil para controles, espacio útil ocupado, etc.) deberá ser el adecuado de acuerdo a las características del usuario objetivo y a las decisiones de diseño tomadas.		

Identificador	RNFI-009	**Nombre**	*Control de pantalla – general*
Prioridad	Alta	**Repercusión**	Alta
Dependencias	RNFD-004, RFND-005, RNFI-007 y RNFI-008	**Fecha**	05/10/14
Descripción		La interacción del usuario con la interfaz, se restringirá a pulsaciones o gestos táctiles en la medida de lo posible.	

Identificador	RNFI-010	**Nombre**	*Navegación de interfaz – general*
Prioridad	Media	**Repercusión**	Baja
Dependencias	RNFR-001, RNFD-004 y RNFI-008	**Fecha**	05/10/14
Descripción		El paso entre diferentes pantallas o sub-pantallas en la interfaz deberá ser fluido e instantáneo (exceptuando animaciones) en la medida de los posible	

Tabla 3.3: Requisitos no funcionales de CleanQuest

3.4 Preproducción – Diseño de implementación y técnico

Una vez completadas las fases de diseño conceptual y artístico y de especificación de requisitos, es posible realizar la fase de diseño clave para el desarrollo de cualquier producto de software, el diseño de implementación y técnico. En esta fase se realizarán las decisiones de diseño relativas a las cuestiones técnicas y de implementación del desarrollo, refiriéndose no sólo a la codificación, sino también a la creación y gestión de *assets* (modelos 2d y 3d, sonido, etc.), estructura y flujo detallados, prototipado, etc.

Estas decisiones de diseño se basan principalmente en la especificación de requisitos del apartado anterior y de forma secundaria en las fases anteriores de diseño conceptual y artístico y concepción. El objetivo de esta fase de diseño de cara al desarrollo es dejar *CleanQuest* lo suficientemente listo para su implementación.

En esta fase de diseño se incluye la realización de un prototipo de bajo de nivel de fidelidad de la aplicación, la selección de herramientas secundarias de cara a la producción y el diseño de la programación.

3.4.1 Prototipo de bajo nivel de fidelidad

Antes de plantear aspectos más técnicos sobre la implementación o el diseño de la programación en general, es necesario presentar un prototipo de baja fidelidad de *CleanQuest*. Se pretende aunar el uso de distintos artefactos de modelado, prescriptivos y de prototipado para elaborar un modelo preliminar del videojuego, esto es, una referencia visual detallada de cada pantalla individual del juego junto con su relación o flujo respecto a las otras pantallas. Igualmente se describen de forma breve pero exacta las interacciones del videojuego con el jugador en cada una de estas pantallas.

El objetivo de este prototipo de bajo nivel de fidelidad es unificar y definir los aspectos principales del videojuego con un grado de abstracción suficiente para poder ser utilizado como **referencia definitiva** en el diseño de implementación y técnico (sobre todo en el diseño de la programación) y de cara a la producción del prototipo software de alto nivel de fidelidad.

Este prototipo de bajo nivel de fidelidad como tal, es clasificado como una combinación de artefactos de prototipado tipo ***sketching***[29] (con un alto nivel de detalle visual) sobre las perspectivas de apariencia, estructura, flujo, comportamiento e interacción; y de tipo ***wireframe***[30] en lo relativo al detalle de la interfaz.

[29] Artefacto de prototipado rápido en papel para bosquejar algunas de las perspectivas de un sistema o producto de software con un grado de detalle moderado.
[30] Artefacto de prototipado que define la estructura, contenido y control de la interfaz de usuario de un sistema o producto de software.

Se utilizará un modelo de tabla para cada pantalla individual con la siguiente estructura o formato:

Nombre: *Nombre de la pantalla*		***ID:* código de identificación de la pantalla.**
Imagen de la pantalla en cuestion con números en color rojo o azul indicándo zonas de interés en la pantalla, utilizados como referencia para la sección de *wireframe* inferior.		
Descripción: Descripción breve de la pantalla.		
Acceso: Descripción breve de cómo acceder a la pantalla.		
Botones/Zonas de interés	***Propósito o funcionalidad***	***Notas***
(nº de la zona de interés en la imagen)	Propósito o funcionalidad de la zona de interés en la imagen	Notas sobre la zona de interés en la imagen

Figura 3.18: Formato de pantalla en el prototipo de baja

- **Nombre.** Nombre narrativo basado en la descripción que identifica a la pantalla.

- **ID.** Código que identifica unívocamente a la pantalla respecto a las demás. El formato de este código es el siguiente: *(Ej)(Sub)CODIGO*. *CODIGO* está formado por una combinación de caracteres alfanuméricos, en mayúsculas, con la posibilidad de añadir un guión (“-“) de separación y un máximo de 8 caracteres. Este *CODIGO* puede ser una abreviación, siglas o cualquier otra cadena simbólica que represente de forma intuitiva el nombre de la pantalla. El prefijo opcional *(Sub)*, indica que la pantalla se trata de una sub-pantalla (Ej. *Prompt*, *pop-up*, etc.) dentro de otra pantalla. El prefijo opcional *(Ej)* indica que la pantalla se trata de un ejemplo representativo dentro de una serie o conjunto de pantallas demasiado similares como para dedicarlas una tabla individual. Este ID se utilizará para referirse a pantallas ajenas dentro de la tabla de una. Ejemplos de este ID pueden ser: *“MP”, “CP-1”, “(Ej)(Sub)CP-ICL”* o *“(Sub)GC-VC”.*

- **Descripción.** Descripción narrativa breve de la pantalla. Hace alusión a su propósito y si procede destaca alguno de sus elementos de cara a su implementación.

- **Acceso.** Indica cómo acceder a la pantalla (Desde que otra pantalla y que acciones son necesarias para llegar).

- **Botones/Zonas de interés.** Sección del prototipo tipo *wireframe* que identifica los botones, áreas interactivas o cualquier otra zona de interés de la imagen superior. Se compone de un número entre paréntesis que coincide con un número reflejado en la imagen superior. Este número puede ser de color rojo, si identifica a una zona con la cual el jugador puede interactuar y/o afecta directamente a la jugabilidad (botones, zonas táctiles, etc.) o color azul, si indica una zona con la que el jugador

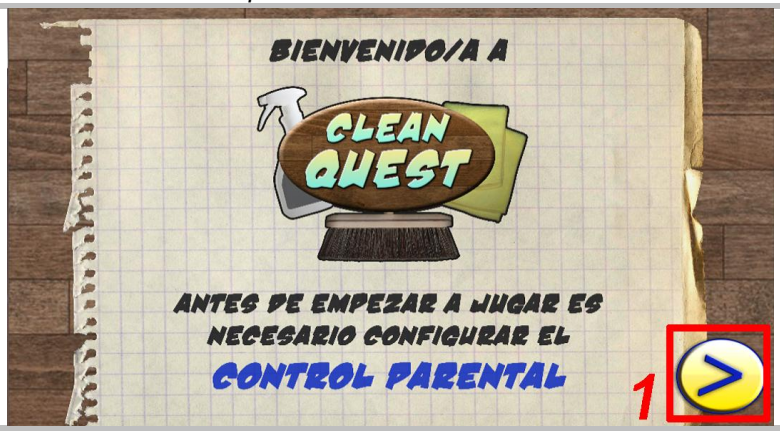
no puede interactuar y/o afecta de forma indirecta a la jugabilidad (contador de tiempo y puntuación, etc.).

- **Propósito.** Descripción del propósito o funcionalidad que tiene el botón o la zona de interés, de cara a su implementación.

- **Notas.** Información adicional sobre el botón o la zona de interés, ya sea condiciones para alcanzar su propósito, funcionalidad extra derivada de su propósito o detalles estéticos de relevancia.

A continuación se presentan cada una de las pantallas del juego, de forma individual o en grupo según proceda y se describen de forma detallada cada uno de los componentes de su interfaz. Finalmente se mostrará un esquema general con el flujo detallado de todo el conjunto de pantallas.

Grupo de pantallas del control parental

Nombre: *1ª Pantalla del control parental – Pantalla de bienvenida*	***ID:*** ***CP-1***
	
Descripción: Primera pantalla del control parental, a modo de bienvenida la primera vez que se inicia el juego. Muestra el logo principal del juego e informa de que se va a proceder a la configuración del control parental.	
Acceso: La primera vez que se inicia la aplicación, es la primera pantalla en aparecer.	

Botones/Zonas de interés	***Propósito o funcionalidad***	***Notas***
(1)	Pasar a *CP-2*	-

Figura 3.19: Pantalla CP-1 del prototipo de baja fidelidad de CleanQuest

Nombre: *2ª Pantalla del control parental – Vídeo introductorio*	**ID:** ***CP-2***

Descripción: Segunda pantalla del control parental. Al llegar por primera vez a ella, se reproduce un video haciendo uso del reproductor externo del SO. En este video se introduce a los padres o tutores del jugador a *CleanQuest*: las características principales del juego, el sentido de los cromos virtuales coleccionables y del desafío de realidad aumentada de cara a resolver el problema propuesto y la necesidad de que ellos introduzcan los códigos/contraseñas y los lugares físicos respectivos para que todo pueda funcionar. Se permite al usuario volver a reproducir este video.

Acceso: Desde *CP-1*, mediante (1); desde *CP-3*, mediante (1).

Botones/Zonas de interés	***Propósito o funcionalidad***	***Notas***
(1)	Pasar a *CP-1*.	-
(2)	Volver a reproducir el vídeo introductorio.	-
(3)	Pasar a *CP-3*.	-

Figura 3.20: Pantalla CP-2 del prototipo de baja fidelidad de CleanQuest

Nombre: *3ª Pantalla del control parental – Introducción de códigos/contraseña y lugares*	**ID:** ***CP-3***

Descripción: Tercera y última pantalla del control parental. Permite a los padres o tutores del jugador establecer los códigos/contraseñas y los lugares físicos respectivos a cada tarea del desafío de realidad aumentada.

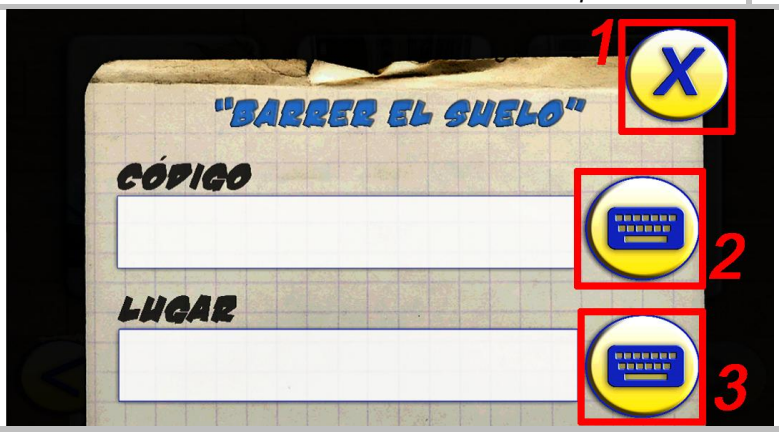
Acceso: Desde *CP-2*, mediante (3); desde *(Ej)(Sub)CP-ICL*, mediante (1)

Botones/Zonas de interés	***Propósito o funcionalidad***	***Notas***
(1)	Pasar a *CP-2*.	-
(2)	Pasar a la *(Ej)(Sub)CP-ICL* correspondiente a la tarea del botón pulsado pulsada. Reflejar si se han introducido bien o mal los datos dentro de la *(Ej)(Sub)CP-ICL* correspondiente.	Abre cada *(Ej)(Sub)CP-ICL* con una animación. Si se introducen mal alguno de los datos en esa sub-pantalla y se cierra, el botón correspondiente a ella reflejará un halo de color rojo alrededor suyo. En el caso contrario (los datos se han introducido correctamente) mostrará un halo de color verde.

(3)	Pasar a *MP*.	Sólo disponible (deja de ser translucido) cuando se hayan introducido correctamente los datos de en cada *(Ej)(Sub)CP-ICL*, esto es, los 3 botones de (2) deben aparecer con un halo verde alrededor suyo.

Figura 3.21: Pantalla CP-3 del prototipo de baja fidelidad de CleanQuest

Nombre: *Prompt de introducción de datos de la 3ª Pantalla del control parental*	***ID:*** ***(Ej)(Sub)CP-ICL***



Descripción: Sub-pantalla o *prompt* para introducir y establecer el código/contraseña de confirmación de cada tarea del desafío de realidad aumentada y el lugar físico del entorno doméstico donde debe ser realizada.

Acceso: Desde CP-3, mediante (2).

Botones/Zonas de interés	***Propósito o funcionalidad***	***Notas***
(1)	Cierra el *prompt*. Pasa a *CP-3*.	Cierra el *prompt* con una animación inversa a la que se reprodujo al abrirlo.
(2)	Abre la herramienta de teclado externa de *Android* para establecer el texto del código/contraseña de confirmación de la tarea del desafío de realidad aumentada correspondiente.	Se aplican restricciones de entrada de texto en tiempo real, como tamaño máximo de cadena de texto o la utilización de cadenas repetidas en otras tareas. Esto se refleja con un texto rojo de aviso en la barra blanca.
(3)	Abre la herramienta de teclado externa de *Android* para establecer el texto del lugar físico deseado del la tarea del desafío de realidad aumentada correspondiente.	Se aplican restricciones de entrada de texto en tiempo real, como tamaño máximo de cadena de texto o la utilización de cadenas repetidas en otras tareas. Esto se refleja con un texto rojo de aviso en la barra blanca.

Figura 3.22: Pantalla (Ej)(Sub)CP-ICL del prototipo de baja fidelidad de CleanQuest

Grupo de pantallas del menú principal

Nombre: *Menú principal*	**ID:** ***MP***

Descripción: Menú principal del juego. Permite el acceso al resto de secciones del juego y es el punto de retorno al salir de cada una de ellas. Diseñado para ser lo más intuitivo posible. Presenta un fondo transparente con un modelo de la habitación de Jaimito 3d de fondo, rotando sobre sí misma lentamente.

Acceso: Desde *CP-3*, mediante (3), la primera vez que se inicial la aplicación. Las subsiguientes veces que se inicie, la aplicación comienza en esta pantalla. Desde *GC*, mediante (1); desde *(Sub)JUG-GEN3* mediante (1); desde *(Ej)ARG*, mediante (1).

Botones/Zonas de interés	***Propósito o funcionalidad***	***Notas***
(1)	Salir del juego. Cierra la aplicación.	-
(2)	Pasa a *(Ej)(Sub)MP-SF*	Abre el *(Ej)(Sub)MP-SF* correspondiente con una animación.
(3)	Pasa a *GC*	-

Figura 3.23: Pantalla MP del prototipo de baja fidelidad de CleanQuest

Nombre: *Sub-pantalla de selección de fases de un mini-juego*	**ID:** ***(Ej)(Sub)MP-SF***

Descripción: Sub-pantalla que permite seleccionar la fase de un mini-juego que jugar. Igualmente muestra el trofeo de mayor valor obtenido en cada una (de forma análoga a una máxima puntuación).

Acceso: Desde CP-3, mediante (3), la primera vez que se inicial la aplicación. Las subsiguientes veces que se inicie, la aplicación comienza en esta pantalla.

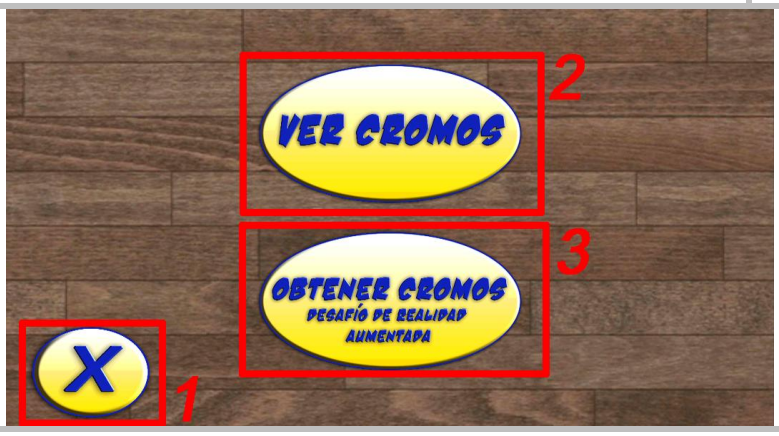
Botones/Zonas de interés	***Propósito o funcionalidad***	***Notas***
(1)	Pasa a una fase (fase 1, 2 o 3) del mini-juego correspondiente, a *(Ej)JUG-GEN1.*	Los botones de las fases a partir de la primera aparecen bloqueados (translucidos) de forma inicial. Para desbloquear cada uno es necesario completar la fase anterior al menos

		una vez (con cualquier trofeo o cromo). Debajo de cada botón se muestra el trofeo de mayor valor obtenido en esa fase actualmente.
(2)	Cierra la sub-pantalla. Pasa a *MP.*	Cierra la sub-pantalla con una animación inversa a la que se reprodujo al abrirlo.

Figura 3.24: Pantalla (Ej)(Sub)MP-SF del prototipo de baja fidelidad de CleanQuest

Grupo de pantallas de la galería de cromos

Nombre: *Galería de cromos*	***ID:*** ***GC***



Descripción: Pantalla principal de la galería de cromos. Permite el acceso al visor de cromos coleccionables obtenidos y a la sub-pantalla de introducción de códigos para el desafío de realidad aumentada.

Acceso: Desde *MP*, mediante (3); desde *(Sub)GC-VC* mediante (2); desde *(Sub)GC-IC* mediante (2).

Botones/Zonas de interés	***Propósito o funcionalidad***	***Notas***
(1)	Cierra la pantalla. Pasa a *MP*	-
(2)	Pasa a *(Sub)GC-VC*	-
(3)	Pasa a *(Sub)GC-IC*	-

Figura 3.25: Pantalla GC del prototipo de baja fidelidad de CleanQuest

Nombre: *Sub-pantalla de visor de cromos de la galería de cromos*	***ID:*** ***(Sub)GC-VC***

Descripción: Sub-pantalla de visor de cromos de la galería de cromos. Permite ver la colección de cromos obtenidos hasta la fecha e información extendida de cada uno de ellos. Más concretamente el nombre y el modelo 3d al que representa.

Acceso: Desde *GC*, mediante (2).

Botones/Zonas	***Propósito o funcionalidad***	***Notas***

de interés		
(1)	Al seleccionar uno de los cromos obtenidos, aparece el nombre del contenido que representa el cromo y su modelo 3d, tal y como aparece en la imagen de ejemplo al haber pulsado el cromo del molino.	El modelo 3d rota sobre sí mismo.
(2)	Pasa a la anterior tanda de cromos de la colección.	Dentro de la misma pantalla.
(3)	Cierra la sub-pantalla. Pasa a *GC*	-
(4)	Pasa a la siguiente tanda de cromos de la colección.	Dentro de la misma pantalla.

Figura 3.26: Pantalla (Sub)GC-VC del prototipo de baja fidelidad de CleanQuest

Nombre: *Sub-pantalla de introducción de códigos/contraseñas de la galería de cromos*	***ID:*** ***(Sub)GC-IC***

Descripción: Sub-pantalla de introducción de códigos/contraseñas de la galería de cromos. Permite la introducción de los códigos necesarios para desbloquear los cromos especiales como recompensas de los desafíos de realidad aumentada una vez que los padres o tutores han confirmado la realización de estas tareas en la vide real en el lugar físico del entorno doméstico establecido.
Acceso: Desde *GC*, mediante (3); Desde *(Ej)(Sub)GC-ICX*, mediante (1).

Botones/Zonas de interés	***Propósito o funcionalidad***	***Notas***
(1)	Abre la herramienta de teclado externa de *Android* para introducir el texto del código/contraseña de confirmación de la tarea del desafío de realidad aumentada correspondiente y desbloquear el cromo. Según si el texto introducido coincide o no con alguno de los códigos introducidos durante el control parental, se pasará a una instancia de *(Ej)(Sub)GC-ICX*	-
(2)	Cierra la sub-pantalla. Pasa a *GC*	-
(3)	Botón para resetear todos los códigos/contraseñas y lugares en cada tarea del desafío de realidad aumentada, en el caso de que se olviden.	Reflejado sólo visualmente, no se pretende su implementación de cara al prototipo de software funcional, ya que no es necesario para las pruebas posteriores.

Figura 3.27: Pantalla (Sub)GC-IC del prototipo de baja fidelidad de CleanQuest

Nombre: *Pop-up de obtención de cromo*	**ID:** ***(Ej)(Sub)GC-ICX***

Descripción: Pop-up que muestra el cromo obtenido al introducir un código de confirmación correcto en *(Sub)GC-IC* o en el caso contrario un mensaje de error.

Acceso: Desde *(Sub)GC-IC*, mediante (1).

Botones/Zonas de interés	Propósito o funcionalidad	Notas
(1)	Al pulsar en cualquier lugar de la pantalla cierra el Pop-up. Pasa a *(Sub)GC-IC.*	-

Figura 3.28: Pantalla (Ej)(Sub)GC-ICX del prototipo de baja fidelidad de CleanQuest

Grupo de pantallas de mini-juegos genéricas

Nombre: *Pantalla de inicio de fase*	**ID:** ***(Ej)JUG-GEN1***

Descripción: Pantalla de inicio de fase de un mini-juego. El familiar de Jaimito correspondiente al mini-juego elegido pone en situación al jugador, explicándole su objetivo dentro de la fase de forma sutil, intercalando fragmentos de trasfondo.

Acceso: Desde *(Ej)(Sub)MP-SF*, mediante (1), esto es, al iniciar una fase de un mini-juego.

Botones/Zonas de interés	Propósito o funcionalidad	Notas
(1)	Detiene la conversación con el familiar de Jaimito y pasa a la fase en sí, esto es, pasa a *(Ej)JUG-RE*, *(Ej)JUG-BA* o *(Ej)JUG-LI* según proceda.	-

Figura 3.29: Pantalla (Ej)JUG-GEN1 del prototipo de baja fidelidad de CleanQuest

Nombre: *Pantalla de final de fase*	**ID:** ***(Ej)JUG-GEN2***

Descripción: Pantalla de final de fase de un mini-juego. El familiar de Jaimito correspondiente al mini-juego elegido muestra y comenta cada premio obtenido al finalizar la fase, ya sean trofeos o cromos coleccionables.

Acceso: Desde *(Ej)JUG-RE*, *(Ej)JUG-BA* o *(Ej)JUG-LI,* al terminar la fase.

Botones/Zonas de interés	Propósito o funcionalidad	Notas
(1)	Al pulsar el botón se muestra el siguiente premio obtenido (sólo en el caso de haber obtenido varios cromos a la vez). Cuando ya no quedan premios por mostrar el botón se torna de color anaranjado y muestra el texto "terminar". Al pulsarlo pasa a la siguiente fase de ese mini-juego o si es la tercera (última fase), pasa a la presentación del desafío de realidad aumentada relativo a ese mini-juego, *(Ej)ARG*	-

Figura 3.30: Pantalla (Ej)JUG-GEN2 del prototipo de baja fidelidad de CleanQuest

Nombre: *Menú de pausa*	**ID:** ***(Sub)JUG-GEN3***

Descripción: Pantalla de pausa del juego. Pausa la acción (impide al jugador el control de la jugabilidad de la fase y congela los contadores de tiempo y puntuación) y permite volver al menú principal.

Acceso: Desde *(Ej)JUG-RE*, *(Ej)JUG-BA* o *(Ej)JUG-LI*, al pulsar los botones universales de los dispositivos con *Android*: *return* o *menú.*

Botones/Zonas de interés	Propósito o funcionalidad	Notas
(1)	Detiene la fase del mini-juego y pasa a *MP*	-
(2)	Cierra la sub-pantalla y reanuda el juego. Pasa a *(Ej)JUG-RE*, *(Ej)JUG-BA* o *(Ej)JUG-LI* según proceda.	-

Figura 3.31: Pantalla (Sub)JUG-GEN3 del prototipo de baja fidelidad de CleanQuest

Grupo de pantallas de jugabilidad en mini-juegos

Nombre: *Ejemplo de jugabilidad en una fase del minijuego "Recoger los juguetes"*	***ID:*** ***(Ej)JUG-RE***

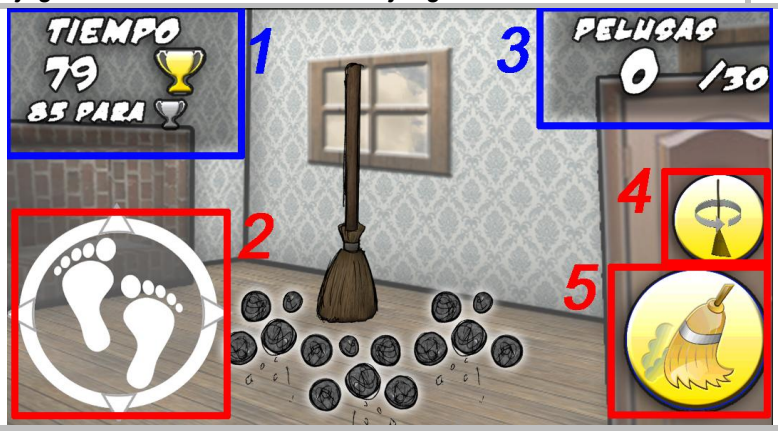
Descripción: Pantalla de ejemplo de una de las fases del mini-juego "Recoger los juguetes". En ella se muestra la interfaz genérica que refleja las reglas comunes de los mini-juegos y los elementos típicos que afectan a la jugabilidad de este mini-juego en concreto. En este caso, los juguetes y el baúl de los juguetes.

Acceso: Desde la *(Ej)JUG-GEN1* correspondiente, mediante (1); desde *(Sub)JUG-GEN3*, mediante (2);

Botones/Zonas de interés	***Propósito o funcionalidad***	***Notas***
(1)	Contador de tiempo progresivo. Muestra el tiempo actual que ha transcurrido desde el comienzo de esta pantalla en segundos. Igualmente muestra el trofeo actual al que se opta en el rango de tiempo actual y el límite para pasar al siguiente rango de tiempo.	No se trata de un botón, es parte de la interfaz que regula la jugabilidad.
(2)	Contador de los hitos o acciones realizadas durante el mini-juego, en este caso, introducir un juguete en el baúl de los juguetes. Muestra el número total necesario para completar la fase. Al realizar el número de hitos requerido, pasa a *(Ej)JUG-GEN2*	No se trata de un botón, es parte de la interfaz que regula la jugabilidad.
(3)	Área de interacción con los elementos de la fase. El jugador debe identificar los juguetes, que se diferencian del resto de los elementos por que les rodea un halo de color amarillo, cogerlos mediante mecanismos *drag and drop* y soltarlos dentro del baúl de los juguetes.	-

Figura 3.32: Pantalla (Ej)JUG-RE del prototipo de baja fidelidad de CleanQuest

Nombre: *Ejemplo de jugabilidad en una fase del minijuego "Barrer el suelo"*	***ID:*** ***(Ej)JUG-BA***



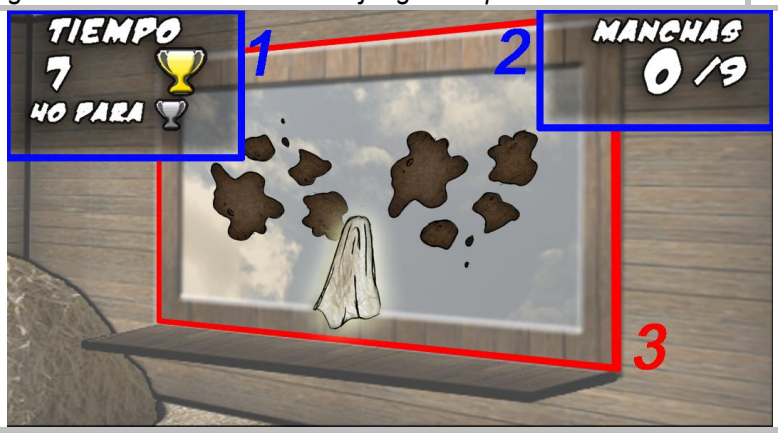
Descripción: Pantalla de ejemplo de una de las fases del mini-juego "Barrer el suelo". En ella se muestra la interfaz

genérica que refleja las reglas comunes de los mini-juegos y los elementos típicos que afectan a la jugabilidad de este mini-juego en concreto. En este caso, la escoba y las pelusas.
Acceso: Desde la *(Ej)JUG-GEN1* correspondiente, mediante (1); desde *(Sub)JUG-GEN3*, mediante (2);

Botones/Zonas de interés	***Propósito o funcionalidad***	***Notas***
(1)	Contador de tiempo progresivo. Muestra el tiempo actual que ha transcurrido desde el comienzo de esta pantalla en segundos. Igualmente muestra el trofeo actual al que se opta en el rango de tiempo actual y el límite para pasar al siguiente rango de tiempo.	No se trata de un botón, es parte de la interfaz que regula la jugabilidad.
(2)	Área de interacción con los elementos de la fase. En este caso un *joystick* táctil utilizado para conducir la escoba por el escenario en tercera persona.	El *joystick* permanece translucido hasta ser usado, volviéndose completamente visible.
(3)	Contador de los hitos o acciones realizadas durante el mini-juego, en este caso, conducir una de las pelusas con la escoba en un rincón indicado. Muestra el número total necesario para completar la fase. Al realizar el número de hitos requerido, pasa a *(Ej)JUG-GEN2.*	No se trata de un botón, es parte de la interfaz que regula la jugabilidad.
(4)	Área de interacción con los elementos de la fase. En este caso un botón para girar 90º la escoba y tener más control sobre ella	El botón permanece translucido hasta ser usado, volviéndose completamente visible.
(5)	Área de interacción con los elementos de la fase. En este caso un botón para agitar la escoba (dar un “escobazo”) en la dirección en la que se esté apuntando con ella (regulable mediante (4)). El objetivo es apartar o lanzar las pelusas que estén en su camino con más fuerza y más rápidamente hacia la dirección deseada, en lugar de conducirlas lentamente.	El botón permanece translucido hasta ser usado, volviéndose completamente visible.

Figura 3.33: Pantalla (Ej)JUG-BA del prototipo de baja fidelidad de CleanQuest

Nombre: *Ejemplo de jugabilidad en una fase del minijuego “Limpiar los cristales”*	***ID:*** ***(Ej)JUG-LI***



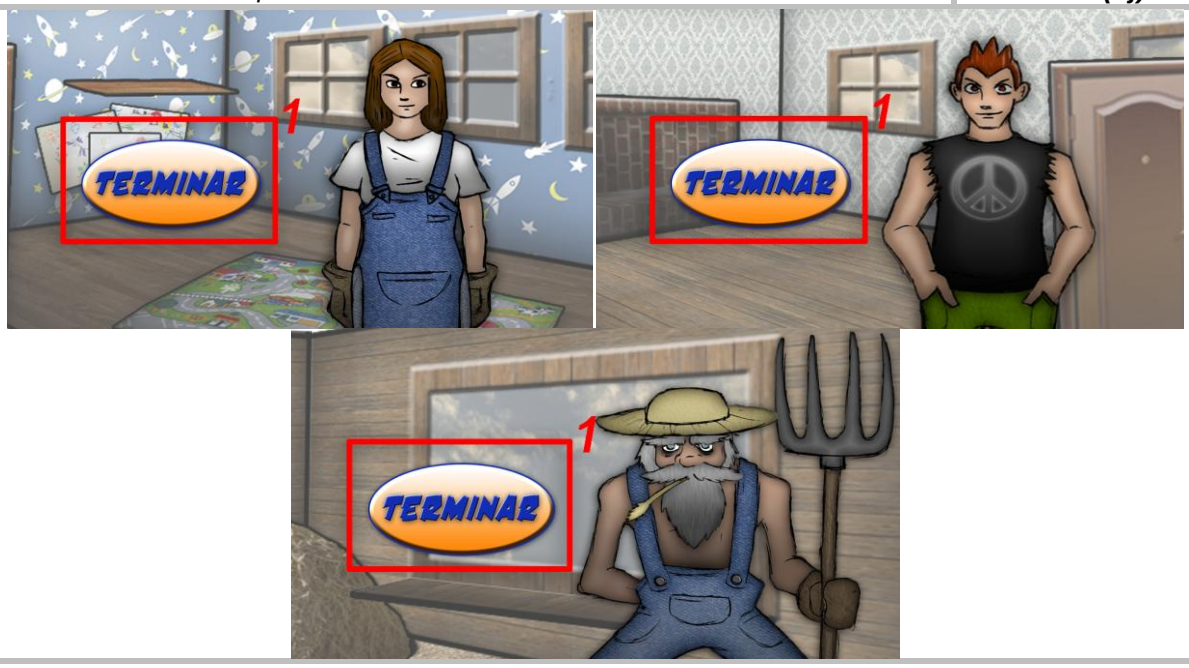
Descripción: Pantalla de ejemplo de una de las fases del mini-juego “Limpiar los cristales”. En ella se muestra la interfaz genérica que refleja las reglas comunes de los mini-juegos y los elementos típicos que afectan a la jugabilidad de este mini-juego en concreto. En este caso, el trapo y las manchas del ventanal.
Acceso: Desde la *(Ej)JUG-GEN1* correspondiente, mediante (1); desde *(Sub)JUG-GEN3*, mediante (2);

Botones/Zonas de interés	***Propósito o funcionalidad***	***Notas***
(1)	Contador de tiempo progresivo. Muestra el tiempo actual que ha transcurrido desde el comienzo de esta pantalla en segundos. Igualmente muestra el trofeo actual al que se opta en el rango de tiempo actual y el límite para pasar al siguiente rango de tiempo.	No se trata de un botón, es parte de la interfaz que regula la jugabilidad.
(2)	Contador de los hitos o acciones realizadas durante el mini-	No se trata de un botón, es parte

	juego, en este caso, desvanecer totalmente una mancha con el trapo. Muestra el número total necesario para completar la fase. Al realizar el número de hitos requerido, pasa a *(Ej)JUG-GEN2.*	de la interfaz que regula la jugabilidad.
(3)	Área de interacción con los elementos de la fase. El jugador debe identificar el trapo, que se diferencian del resto de los elementos por que les rodea un halo de color amarillo, cogerlo mediante mecanismos *drag and drop* y restregarlo contra las manchas del cristal hasta desvanecerlas.	-

Figura 3.34: Pantalla (Ej)JUG-LI del prototipo de baja fidelidad de CleanQuest

Pantalla de presentación del desafío de realidad aumentada

Nombre: *Pantalla de presentación del desafío de realidad aumentada*	**ID: (Ej)ARG**
	
Descripción: Pantalla de presentación del desafío de realidad aumentada, para cada mini-juego. Al completar las 3 fases de un mini-juego, aparecerá esta pantalla, donde el familiar de Jaimito que ha asistido al mini-juego se dirigirá directamente al jugador (rompiendo el "cuarto muro"). Le explica en qué consiste el desafío de realidad aumentada relativo a la tarea que representa el mini-juego que acaba de completar y le invita a participar en el mismo.	
Acceso: Desde *(Ej)JUG-GEN2,* mediante (1) al terminar todos los premios y si es el final de la tercera fase.	

Botones/Zonas de interés	Propósito o funcionalidad	Notas
(1)	Detiene la conversación si se está realizando. Pasa a *MP*	-

Figura 3.35: Pantalla (Ej)ARG del prototipo de baja fidelidad de CleanQuest

Diagrama de flujo y relación de pantallas

A continuación se presenta un diagrama que muestra el flujo y la relación entre las pantallas anteriores. Se reflejará la correspondencia de los grupos de pantallas y, en cuanto a las pantallas de ejemplo ("(Ej)"), se especificará de forma textual cada instancia de las mismas

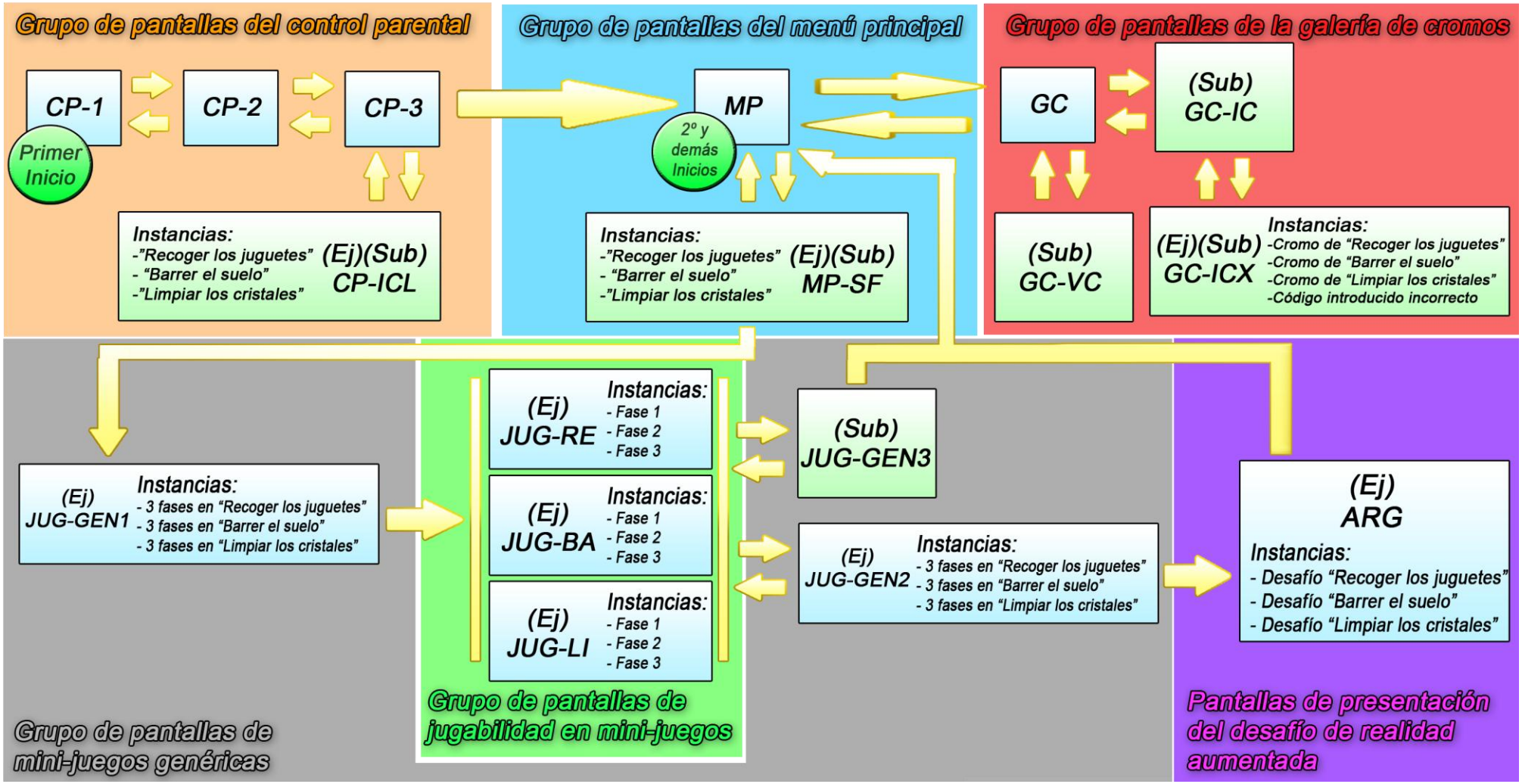


Figura 3.36: Diagrama de flujo detallado de CleanQuest

3.4.2 Selección de herramientas secundarias

Además de *Unity*, kit de desarrollo principal, es necesario establecer una serie de herramientas secundarias para el desarrollo. Con estas herramientas se implementarán de forma externa los *assets* que no puedan o que no sean adecuados para ser creados directamente en *Unity* y que posteriormente sean gestionados por el mismo.

Para su selección se obvia una comparativa compleja como la realizada en el análisis del estado del arte y en su lugar, se atiende principalmente a dos motivos: su relación calidad-precio y la disponibilidad de experiencia previa con los mismos. De esta forma se seleccionan los siguientes programas como herramientas secundarias para el desarrollo, según su propósito dentro del mismo:

- **Codificación de scripts.** Se elige el entorno de desarrollo integrado de *Unity*, *MonoDevelop-Unity*. Se opta por éste frente a otras alternativas debido a que para su propósito (escribir código, compilarlo y depurar los errores) es suficiente.

- **Modelado y animación 3d.** Se eligen por un lado, el programa *Sculptris Alpha 6* para el modelado y texturización de personajes y objetos de estilo orgánico (como los familiares de Jaimito) y *Blender* como programa de modelado auxiliar, animación y composición. Ambos programas son gratuitos, con una calidad que rivaliza a la de análogos de pago y además se dispone de gran experiencia previa con ellos.

- **Modelado y composición 2d.** Se elige el programa *Adobe Photoshop CS5*. Este flexible programa, ya utilizado previamente en la maquetación y realización de esta memoria, aporta todo tipo de herramientas de edición gráficas para modelar la interfaz de usuario 2d y ayudar al texturizado en el modelado 3d. Su elección se basa en que, aunque es de pago, se dispone de una licencia y experiencia previa.

- **Sonido.** Se elige el programa de edición de audio *Audacity* por ser gratuito, su facilidad de uso y por qué se dispone de gran cantidad de experiencia previa con él.

- **Otros.** Para otras tareas extra como la edición de video (para el vídeo de introducción en el control parental), se elige el programa de edición de video *Adobe After Effects CS5* y el capturador de pantalla *Fraps*. Ambos programas presentan una gran competencia en su propósito. *Fraps* es gratuito y sencillo de utilizar. Sobre *After Effects,* se contaba con experiencia previa y una licencia del producto.

Debido a la elección de estos programas, previamente familiares y con un grado de experiencia inicial alto en cada uno de ellos, el proceso de producción se verá acelerado en gran medida.

3.4.3 Diseño de la programación

Una vez establecidos la especificación de los requisitos para desarrollo y elaborado un prototipo de bajo nivel de fidelidad, se posee una referencia prácticamente definitiva, tanto narrativa como visual, para poder plantear un diseño para la programación de cara a la producción.

El problema es que un proyecto o videojuego en *Unity* no se compone únicamente de clases o ficheros de código de programación, sino que también influyen en gran medida los parámetros y componentes gestionados por el propio editor de escena de *Unity* (sobre todo al tratarse de un videojuego no muy complejo como éste). Por este motivo, no es posible reducir todo el diseño de la programación a diagramas de clases o secuencia convencionales. En su lugar, es necesario abstraer la mayor parte de estos aspectos en lo que respecta al editor de *Unity* y elegir los ficheros de código o *scripts* más representativos para garantizar un diseño de la programación entendible y coherente de cara a la producción.

De esta manera para el diseño de la programación se propone, primeramente, realizar un **diseño de *assets***, esto es, un diseño de cómo integrar los *assets* (recursos como modelos 3d, 2d, sonidos, etc.) en el juego por medio del editor de *Unity* y a continuación, presentar unos **diagramas de clases y secuencia** sobre los scripts o grupos de scripts más relevantes de cara al desarrollo.

3.4.3.1 Diseño de *assets*

El diseño de *assets* tiene por objetivo establecer la forma en la que los *assets* son integrados por el editor de *Unity* dentro de una escena en el videojuego e identificar los componentes de los mismos que necesitan ser implementados mediante las herramientas secundarias comentadas anteriormente. Para ello, primeramente es necesario identificar los tipos de *assets* principales que van a ser utilizados durante el desarrollo y sus respectivos componentes.

Una vez estudiado el editor de *Unity* en profundidad (y haciendo uso de la experiencia inicial que se poseía del mismo), se pueden extraer los siguientes **componentes de *asset*** más relevantes que afectarán de forma directa al desarrollo de los tipos *assets*:

-  **– *Animation*.** Otorga al *asset* una animación o conjunto de animaciones reproducibles realizadas con un programa externo.

-  **– *Audio Source*.** Otorga al *asset* un emisor de audio en la escena, ya sea música, efectos de sonido o clips de voz.

-  **– *Camera*.** Otorga al *asset* una vista principal de la escena, esto es, rango de visión que tiene el usuario de todos los elementos de la escena (sólo puede haber uno al mismo tiempo en la escena).

-  **– *Character Controller*.** Utilizado en ciertos scripts y el editor para diferenciar al *asset* con privilegios y funcionalidades del jugador del resto.

-  **– *Collider*.** Otorga al *asset* un envoltorio transparente que le permite interactuar con los demás *assets* "físicamente" en el entorno virtual. Permite reconocer cuando un *asset* colisiona o entra en contacto con otro dentro de la escena y responde a esa colisión, ya sea como una barrera estableciendo un límite con otros *assets* o desencadenado otro tipo de evento.

-   **– *GUIText/GUITexture*.** Otorga al *asset* un texto o textura (imagen) editable que compone la interfaz de usuario. Sólo visible por la *Camera*.

-  **– *Material*.** Combinación de textura y *shader*[31] que envuelve al modelo 3d (*mesh*) de un *asset*. En otras palabras, da color y sombreado al modelo 3d.

-  **– *Mesh Renderer*.** Indica al editor renderizar (hacer visible) el modelo 3d del *asset* (*mesh*), junto con su *Material* adicionado a ojos del editor y de la *Camera*.

-  **– *Rigid Body*.** Indica al editor que el modelo 3d del *asset* que lo incorpore tiene características físicas coherentes (fuerzas como la gravedad, peso, masa, etc. actúan sobre él) o modificadas a partir de las coherentes. Igualmente interactúa de forma coherente con otros *asset* con *Rigid Body* o *Colliders* (ej. Dos cuerpos que chocan o rozan).

-   **– *Script*.** Liga un fichero que contiene código *JavaScript* o *C#* (lenguajes propuestos para la programación) a un *asset*. Esto le confiere un comportamiento particular en la escena, con la posibilidad de modificar sus propias características o funcionalidades o las de otros *assets* dentro de la escena.

-  **– *Transform*.** Parámetros espaciales y variables del *asset* dentro de la escena. Concretamente posición, rotación y tamaño.

Con todos los posibles componentes de un *asset* considerados, es posible establecer los **tipos de *assets*** que va a ser necesario implementar en el desarrollo a partir de ellos. Éstos se muestran a continuación mediante tablas individuales que recogen el nombre del tipo de *asset*, una breve descripción general, sus componentes predeterminados y la justificación de la inclusión de cada uno de ellos.

[31] En este contexto un *shader* es un procedimiento (unidad de programación) que indica el tipo de sombreado e iluminación que va a tener un modelo (vértice o pixel) durante su renderizado.

Nombre del tipo de asset: ***Fuente de audio***	
Descripción: *Asset* vacío (no se representa visualmente en 3d en la escena) con un único componente, un Audio Source. El propósito de la fuente de audio es reproducir archivos de audio (músicas, efectos de sonido o clips de voz) en la escena, al ser llamadas mediante scripts de otros *assets* o mediante el editor.	
Componentes	**Justificación del uso**
 ***– Audio Source.***	Se utiliza debido a que es la única forma de emitir audio en la escena. Su uso en el *asset* de esta manera, esto es, un *asset* vacío con un único componente *Audio Source*, es debido a que de cara a la codificación es más sencillo tener los emisores de audio en *assets* separados y estáticos, esto es, que no vayan a cambiar con el tiempo. De esta forma se puede reproducir el audio de estos objetos llamándolos desde otros scripts o el editor, sin preocupación por que el objeto vaya a destruirse, desactivarse o su comportamiento vaya a cambiar, interrumpiendo de forma no deseada la reproducción del sonido. Cabe destacar que la utilización de este tipo de *asset* sólo tiene sentido cuando el audio que reproduce no es 3d, esto es, no depende del posicionamiento del emisor (ej. Necesidad de oir un sonido más cercano o lejano al jugador), como son la totalidad de los audios de *CleanQuest*.

Nombre del tipo de asset: ***Fragmento de interfaz de usuario***	
Descripción: *Asset* vacío (no se representa visualmente en 3d en la escena) con componentes 2d relativos a la interfaz. Un conjunto de ellos forma una interfaz de usuario de una pantalla. Pueden tener *Scripts* adheridos que otorgan funcionalidad a la interfaz.	
Componentes	**Justificación del uso**
 ***– GUIText/GUITexture.***	Utilizado para mostrar fragmentos de interfaz de usuario 2d modificables en la pantalla, visibles desde la *Camera*. Se utilizan en fragmentos en vez de uan interfaz completa, debido a que de cara a la codificación es más sencillo considerar cada fragmento o componente de la interfaz de forma individual (ej. Reconocimiento de si el usuario ha pulsado el área de la pantalla táctil perteneciente a un botón de esta interfaz).
 ***– Script.*** **(Opcional)**	Adheridos a algunas partes de la interfaz para otorgarlas funcionalidades extras de bajo nivel no reflejadas en los Scripts *assets* controladores. Usualmente utilizados para reproducir animaciones de entrada o salida en pantalla (como los *Pop-up* o los *Prompt*).
 ***– Transform.***	Necesario para establecer su situación correcta en pantalla.

Nombre del tipo de asset: ***Personaje no jugable***	
Descripción: *Asset* que representa el modelo 3d de los personajes no jugables (familiares de Jaimito, animales de la granja, etc.). Contiene sus animaciones, que serán reproducidas al ser llamadas por los scripts del *asset* controlador de escena.	
Componentes	**Justificación del uso**
 ***– Mesh Renderer.***	Utilizado para que tanto el editor como la cámara puedan ver el modelo 3d del personaje (*mesh*) y se pueda volver visible o invisible según convenga mediante *Scripts* de otros *assets*.
 ***– Animation.***	Animación o conjunto de animaciones del personaje. Permiten aplicar las animaciones realizadas con un programa externo (en este caso blender) a los personajes. Pueden ser llamadas desde *Scripts* de otros *assets* para ser reproducidas.
 ***– Material.***	Utilizado para aplicar las texturas coloridas y los *shader* estilo *cell shading* propuestos durante el diseño conceptual y artístico a los personajes.

 **– *Transform*.**	Necesario para establecer su situación correcta en pantalla.

***Nombre del tipo de asset:* Escenarios y objetos decorativos o que afectan a la jugabilidad de forma pasiva**	
Descripción: *Asset* que representa el modelo 3d de los escenarios y objetos decorativos o que afectan a la jugabilidad de forma indirecta. Según su características pueden ser visibles en no (barreras invisibles), poseer *Colliders* o presentar algún tipo de animación usualmente realizada con *Scripts*.	
Componentes	***Justificación del uso***
 **– *Mesh Renderer*. (Opcional)**	Utilizado para que tanto el editor como la cámara puedan ver el modelo 3d del objeto o escenaio (*mesh*) y se pueda volver visible o invisible según convenga mediante *Scripts* de otros *assets*. Los *assets* que simplemente regulen la jugabilidad pero que no necesiten ser visibles (como las barreras invisibles que limitan el espacio de interacción), no lo utilizan.
 **– *Collider*. (Opcional)**	Sólo utilizado por los objetos que afectan a la jugabilidad de forma indirecta, ya sean obstáculos que agreguen dificultad a la jugabilidad (sillas, mesas, etc.) o barreras limitadoras del espacio de interacción como las barreras invisibles.
 **– *Material*. (opcional)**	Utilizado para aplicar las texturas coloridas y los *shader* estilo *cell shading* propuestos durante el diseño conceptual y artístico a los objetos y escenario. Sólo utilizado en los objetos visibles.
 **– *Script*. (Opcional)**	Utilizados en ciertos objetos para otorgarles animaciones simples programáticas (como la rotación de las aspas del molino) o comportamientos especiales respecto a otros objetos (ej. Barreras invisibles que rechazan los objetos con los que colisionan como una fuerza contraria inusual, como las propuestas para ser utilizadas en el mini-juego de “Recoger los juguetes”.
 **– *Transform*.**	Necesario para establecer su situación correcta en pantalla.

***Nombre del tipo de asset:* Objetos que afectan a la jugabilidad de forma activa**	
Descripción: *Asset* que representa el modelo 3d de los objetos que afectan a la jugabilidad de forma directa. En concreto se trata de los juguetes y el baúl de los juguetes del mini-juego “Recoger los juguetes”, las pelusas, la escoba y el rincón para depositar las pelusas del mini-juego “Barrer el suelo” y las manchas de barro y el trapo del mini-juego “Limpiar los cristales”. Estos *assets* se pueden dividir a su vez en dos sub-grupos, los activos, como los juguetes, la escoba, las pelusas y el trapo, objetos que el jugador controla directamente y los pasivos, como el baúl de los juguetes o el rincón en el que depositar las pelusas, que simplemente esperan a que el jugador realice la acción necesaria con los *assets* activos y desencadenan los eventos necesarios para que el juego pueda proseguir (desactivar objetos, aumentar la puntuación, etc.).	
Componentes	***Justificación del uso***
 **– *Mesh Renderer*.**	Utilizado para que tanto el editor como la cámara puedan ver el modelo 3d del objeto (*mesh*) y se pueda volver visible o invisible según convenga mediante *Scripts* de otros *assets*.
 **– *Collider*.**	Utilizado para que los objetos puedan interactuar entre ellos para garantizar características físicas coherentes (chocar de forma “natural”) o de forma que ciertos objetos reconozcan si un objeto jugable a entrado en contacto con ellos (ej. Necesario para que el baúl de los juguetes reconozca si un juguete ha sido introducido en él).
 **– *Material*.**	Utilizado para aplicar las texturas coloridas y los *shader* estilo *cell shading* propuestos durante el diseño conceptual y artístico a los objetos.
 **– *Rigid Body***	Aporta a los objetos características físicas coherentes simulando que fuerzas como la gravedad, peso, masa, etc. actúan sobre ellos. Utilizado en los objetos activos que el jugador controla directamente, como los juguetes,

	las pelusas, la escoba, etc.
 **– *Animation*. (Opcional)**	Animación o conjunto de animaciones del personaje. Permiten aplicar las animaciones realizadas con un programa externo (en este caso blender) a los personajes. Pueden ser llamadas desde *Scripts* de otros *assets* para ser reproducidas. En este caso animaciones como el embate de la escoba ("escobazo") o el traqueteo del baúl de los juguetes al introducir un juguete en él.
 **– *Script*. (Opcional)**	Utilizados sobre todo en los objetos pasivos para desencadenar los eventos necesarios (reproducir animaciones, control de puntuación de hitos, etc.) y añadir control extra sobre los *asset* tipo jugador
 **– *Transform*.**	Necesario para establecer su situación correcta en pantalla.

Nombre del tipo de asset: ***Jugador***	
Descripción: *Asset* vacío (no se representa visualmente en 3d en la escena) que representa al jugador en la escena jugable, esto es, lo que ve y su funcionalidad de interacción directa con los objetos dentro de los mini-juegos.	
Componentes	***Justificación del uso***
 **– *Character Controller***	Utilizado en los scripts del jugador para diferenciar al *asset* con privilegios y funcionalidades del jugador del resto.
 **– *Camera***	Utilizado para otorgar al jugador la visión de la pantalla y la interfaz de usuario. Igualmente se utiliza como referencia en los *Scripts* del jugador de mini-juegos para reconocer el rango de visión del jugador.
 **– *Script*.**	Controlan la interacción directa del jugador con el resto de *assets* en el caso de los mini-juegos. La interacción con el resto de las interfaces y menús, es llevada a cabo por los Scripts de los *asset* controladores.
 **– *Transform*.**	Necesario para establecer su situación correcta en pantalla.

Nombre del tipo de asset: ***Controladores***	
Descripción: *Asset* vacío (no se representa visualmente en 3d en la escena) con un único componente de Script que gestiona la totalidad de los eventos principales en cada pantalla necesarios para preparar cada escena y garantizar que el jugador (junto con sus Scripts y los Scripts de otros *assets*) menores puedan funcionar y seguir su curso de manera correcta.	
Componentes	***Justificación del uso***
 **– *Script*.**	Hacen la función de script globales que gestionan todos los *assets* en cada pantalla y desencadenan y gestionan los eventos necesarios para garantizar el flujo correcto de cada pantalla y que el resto de los Scripts de otros *assets* puedan funcionar correctamente. Entre otras cosas se encargan del reconocimiento de pulsaciones de botones y del paso entre pantallas. Se ha decidido centralizar de esta manera la gestión de *assets* para facilitar el desarrollo, para evitar contar con un alto número de Scripts menores interrelacionados, más difíciles de manejar/gestionar de cara a la producción.

Tabla 3.4: Tipos de assets en CleanQuest

3.4.3.2 Diagramas de clases

Una vez establecidos los tipos de *assets* principales que van a usarse en el editor de *Unity*, es posible identificar cuáles de ellos van a necesitar ***Scripts*** (componente *Script* del apartado anterior) y extraer los más relevantes y representativos para elaborar **modelos o arquetipos de clases** con una estructura definida, de cara a la programación en la fase de producción.

Antes de ello es necesario hacer un breve inciso, comentando la estructura y dinámica general de los *Scripts* en *Unity*. Aunque *Unity* soporta los lenguajes *JavaScript*, *C#* y *Boo*, para este inciso y en los posteriores diagramas de clase y **se utilizará sólo el lenguaje *JavaScript*** ya que es el lenguaje del que se dispone más experiencia inicial.

Estructura y dinámica general de los Scripts en Unity

Los *Scripts* en *Unity* siguen una estructura y dinámica similar a las típicamente utilizadas en las clases java de *Android*, con métodos o funciones especializadas que actúan según la fase del ciclo de vida del *Script* que esté sucediendo durante la ejecución. Igualmente poseen cualidades de tiempo real, como una función principal (*Update()*) que se ejecuta cada iteración o "frame" de la ejecución, pudiéndose utilizar para implementar funcionalidades constantes o para realizar comprobaciones constantes en la escena.

La estructura y dinámica concreta que presenta un *Script* típico en *Unity*, desglosada en parámetros (variables) y métodos, **obviando los que no resulten útiles** de cara a la implementación de las competencias de los tipos de *Scripts* planteados anteriormente, es la siguiente:

- **Parámetros o variables globales.** Variables globales clásicas. Las variables en los *Scripts* de *Unity* pueden albergar tanto tipos primitivos y objetos convencionales, como referencias a objetos encapsulados más complejos ligados a la escena y al editor, por ejemplo los propios *assets* en la escena (*GameObject*) o sus componentes (*Texture2d*, *Audio*, etc.). Mediante las variables globales, un *Script* puede comunicarse con otro, dentro de la misma escena o pantalla o entre pantallas si se hace persistente.

- **Método *Awake().*** Método general ejecutado antes de la ejecución real de forma aislada, esto es, antes de que la pantalla cargue por completo y sea visible al jugador. Se utiliza para realizar los preparativos de la escena antes de que el *Script* sea sensible al paso del tiempo (empiecen a ejecutar las funciones *Start()* y *Update()*).

- **Método *Start().*** Método general ejecutado en el primer *frame* de la ejecución general de la escena. Utilizado para realizar los preparativos necesarios antes de la ejecución del método *Update()*, de forma similar al método *Awake()*, pero siendo sensible al paso del tiempo, compartiendo la ejecución con los demás *Scripts* o corrutinas.

- **Corrutinas.** Métodos secundarios personalizables con parámetros de entrada y salida que se ejecutan de forma paralela a los métodos generales *Update()* y *Start()*. Son llamados desde los mismos para realizar tareas asíncronas que no pueden realizar ellos mismos, debido a su naturaleza síncrona o tareas que no deban realizar ellos para evitar sobrecarga de código.

- **Métodos ligados a componentes, activados por eventos predeterminados.** Métodos como *OnCollisionEnter(), OnGUI(), OnClick(),* etc. Estos métodos están ligados a ciertos componentes especializados de los *assets* y heredan sus propiedades en forma de parámetros de entrada. Por este motivo deben incluirse en *Scripts* ya adheridos a los *assets* que además contienen ese componente. Se ejecutan de forma paralela a los métodos generales *Update()* y *Start()* y se activan sólo cuando el componente del *asset* al que están ligados cumple ciertas condiciones en la escena. Por ejemplo, el método *OnCollisionEnter()* sólo puede ser incluido en *Scripts* adheridos a un *asset* que además tenga un componente *Collider*, y sólo se activa cuando otro objeto o *asset* en la escena entra dentro del espacio ocupado por el *Collider*. En otras palabras, responde a las colisiones de otros objetos con el que lleva adherido el *Script* con la función. En el caso de *CleanQuest* este método en particular se utiliza, por ejemplo, en el baúl de los juguetes para detectar cuando un juguete es introducido en él.

Diagramas de los Scripts arquetípicos

Para identificar los *Scripts* más relevantes, se descartan los más triviales y de menor tamaño, como los *Scripts* de animaciones, funcionalidades o propiedades muy específicas (rechazo de *Rigid Bodies* por una barrera invisible) que presentan ciertos *assets*. De esta forma se consigue identificar los siguientes arquetipos de *Scripts* más relevantes, presentados en forma de **diagramas UML aislados**, de cara a la codificación:

Asset tipo "Controladores" – Script "Controlador de interfaz"

Script general utilizado para controlar y gestionar la navegación dentro de una misma pantalla y entre pantallas que contengan menús. Esto es, garantiza la interacción correcta del usuario con las interfaces de las pantallas con menús; más concretamente y atendiendo al prototipo de bajo nivel de fidelidad, las pantallas de los grupos de pantallas del control parental, del menú principal y de la galería de cromos. Cada pantalla de estos grupos (incluyendo sub-pantallas) presenta uno de estos *asset* con su *Script* "Controlador de interfaz".

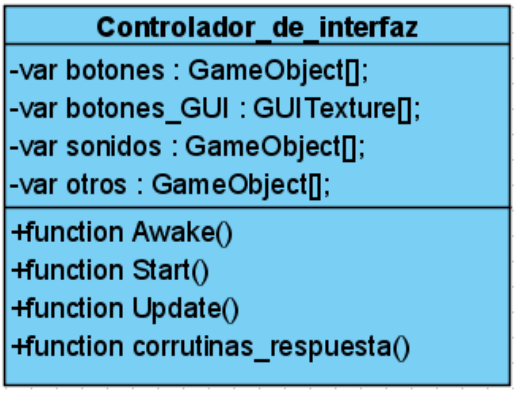


Figura 3.37: Diagrama del arquetipo de Script "Controlador de interfaz"

Los parámetros globales *var botones* y *var botones_GUI* contienen referencias a los *assets* tipo "Fragmentos de la interfaz de usuario" que sean áreas táctiles (botones) y a sus componentes *GUITexture,* respectivamente. *Var sonidos* referencia a todos los *assets* tipo "Fuente de audio" en la escena (utilizados por los métodos a la hora de reproducir un sonido) y *var otros* referencia a otros *assets* más especializados para alguna pantalla o menú en particular.

En la función *Awake(),* se extraen los componentes *GUITexture* de *var botones* y se introducen en *var botones_GUI*. Esto es debido a que, a diferencia de los propios *assets* (*GameObjets*), no es posible referenciar directamente sus componentes al declararlos en el código, a través del editor. Es necesario referenciar estos componentes *GUITexture* por separado ya que en el método *Update()* se utilizan para comprobar en cada *frame* de la ejecución si el usuario ha pulsado alguno de ellos a través de la pantalla táctil (mediante el método nativo *HitTest(),* básicamente se detecta la pulsación de un botón en la pantalla táctil), y responder, llamando a alguna corrutina específica, de la serie de corrutinas *corrutinas_respuesta()*. Estas corrutinas, contienen la funcionalidad de cada uno de las áreas táctiles o botones de la interfaz de usuario.

Asset tipo "Controladores" – Script "Controlador de fase"

Script general utilizado para controlar y gestionar la interfaz y el transcurso de la acción dentro de una fase de un mini-juego. Esto es, garantiza el flujo de pantallas correcto en una fase de un mini-juego (inicio de fase, fase jugable y final de fase) y el flujo de eventos dentro de las mismas (reproducción de animaciones y clips de audio de los familiares de Jaimito, gestión de premios obtenidos, etc.). Lo único que no controla este tipo de *Script* son el cumplimiento de las reglas dentro de la fase jugable, esto es, el paso del tiempo (contador de tiempo progresivo) y el control de puntuación (realización de hitos), ya que es relegado al tipo de *Script* "Controlador de reglas" siguiente. Aun así se comunica con él, a la hora de realizar la gestión de premios en la parte final de la fase. Cada fase de cada mini-juego cuenta con un *asset* tipo controlador con un único *Script* controlador de fase.

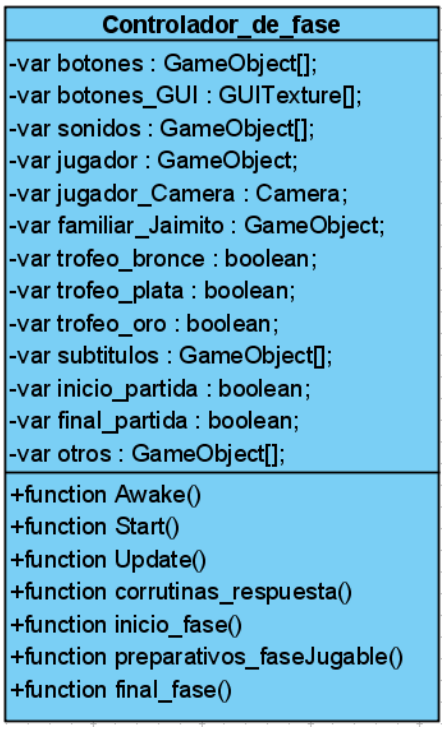


Figura 3.38: Diagrama del arquetipo de Script "Controlador de fase"

Los parámetros globales *var botones* y *var botones_GUI* son similares a sus análogos en el *Script* "Controlador de interfaz", siendo utilizados igualmente por los métodos generales para el reconocimiento de la pulsación de botones táctiles, junto con las *corrutinas_respuesta()*.

Var jugador y *var jugador_Camera* referencian al *asset* tipo "Jugador" y a su componente *Camera* respectivamente. Es necesario utilizar *jugador_Camera* separado para poder acceder a la *Camera* y modificar su posición y ángulo según si la ejecución se encuentra en el inicio de fase, la fase jugable o el final de la fase.

Var familiar_Jaimito referencia al *asset* tipo "Personaje no jugable" de la fase, el familiar de Jaimito que asiste a ella. Contiene todos los componentes y animaciones necesarias que pueden ser llamados desde otros métodos.

Var trofeo_bronce, plata y *oro*, representan el trofeo obtenido por el usuario al completar la fase. Sus valores los pasados por el *Script* "Controlador de reglas" (comentado más adelante) una vez finaliza la fase jugable y se procede al final de fase, para poder realizar la gestión de premios.

Var inicio_partida y *final_partida* se utilizan como indicadores para que el método *Update()* actúe según la parte de la fase en la que se encuentre: inicio de fase, fase jugable o final de fase.

Respecto a los métodos, primeramente, en *Awake(),* se inicializan y preparan todos los parámetros globales de forma correcta. A continuación *Start()* llama a la corrutina

inicio_fase, que ejecuta todos los eventos necesarios para reproducir la *cutscene* de inicio de fase (conversación con el familiar de Jaimito).

Update() por su parte se encuentra fragmentado por condicionales dependientes de las *var inicio_partida* y *final_partida*. Según sus valores el método comprueba distintos eventos durante la ejecución. Durante el inicio, comprueba las pulsaciones del botón táctil para empezar la fase jugable y al detectar una, llama a la corrutina *preparativos_faseJugable()*, que se encarga de preparar la escena para la parte jugable en la fase. Durante la fase jugable, comprueba constantemente si ya se ha acabado la misma, para llamar a la corrutina *final_fase()*, que ejecuta todos los eventos necesarios (gestión de premios, comprobación de cromos guardados en memoria, etc.) para reproducir la *cutscene* de final de fase.

Asset tipo "Objetos que afectan a la jugabilidad de forma activa" – Script "Controlador de reglas".

Script general, adherido a objetos pasivos dentro de la escena, más concretamente, al baúl de los juguetes en el mini-juego "Recoger los juguetes", el rincón donde depositar las pelusas en el mini-juego "Barrer el suelo" y el conjunto de las manchas en el mini-juego "Limpiar los cristales". Se encarga de controlar las reglas comunes dentro de una fase de un mini-juego, esto es, llevar el tiempo transcurrido en segundos y reflejarlo en el contador progresivo de la interfaz. Igualmente se encarga de reconocer el hito/acción numerable cuando lo realiza el jugador (ej. Introducir un juguete en el baúl de los juguetes) y lo refleja en la puntuación de la interfaz. Cuando el jugador consigue realizar todos los hitos necesarios en una fase, el *Script* se encarga de acabar la fase jugable, pasar a la sub-pantalla de final de fase y pasarle el tiempo final obtenido por el jugador en la fase al *Script* "Controlador de fase" para que pueda realizar la gestión de premios correctamente.

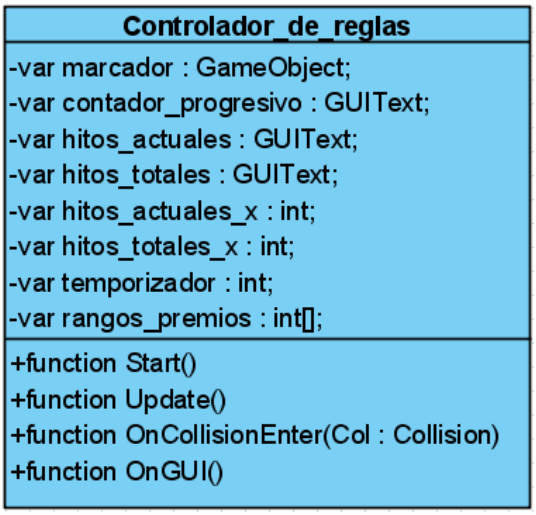


Figura 3.39: Diagrama del arquetipo de Script "Controlador de reglas"

Los parámetros globales *var contador_progresivo* e *hitos_actuales* (junto con *hitos_totales*) referencian a los *assets* tipo "Fragmento de interfaz de usuario" relativos al contador progresivo y al contador de hitos o acciones numerables respectivamente,

en la fase jugable. Antes de asignarles un valor numérico para mostrar por pantalla, se opera primero de forma temporal con los enteros auxiliares *hitos_actuales_x, hitos_totales_x* y *temporizador*. El *array* de valores enteros *rangos_premios* se utiliza como referencia para calcular la relación entre los rangos de tiempo y los premios a los que se opta en esos rangos de tiempo para una fase en particular.

En el método general *Start()*, se reflejan los valores iniciales en *contador_progresivo* (0), *hitos_actuales* (0) e *hitos_totales* (particulares de esa fase). Igualmente se inicializan los valores enteros auxiliares. En el método Update(), por su parte, se comprueba a cada momento el paso del tiempo en segundos y se incrementa el contador auxiliar *temporizador* cada vez que pasa un segundo.

El método ligado a componente *OnCollisionEnter()* se activa cada vez que el usuario realiza un hito o acción numerable en esa escena. Cuando esto sucede, el método aumenta el contador auxiliar *hitos_actuales_x*.

El método ligado a componente *OnGUI()* se activa cada vez que un valor cuya referencia la tiene un componente de interfaz (GUIText o GUITexture) cambia de valor. En este caso concreto el método se activa cada vez que *hitos_actuales_x* o *temporizador* incrementan su valor. Cada vez que se activa asigna los nuevos valores auxiliares a los *GUIText hitos_actuales* y *contador_progresivo* respectivamente para reflejarlos visualmente en la interfaz, esto es, para que los contadores de tiempo y de hitos en la interfaz avancen visualmente. Igualmente, cada vez que sucede esto, el método comprueba si el número de hitos realizados por el jugador se corresponde con el número de hitos totales (*hitos_actuales_x = hitos_totales_x*) y si se da el caso, significa que el usuario a completado la fase. En ese caso el método recoge el valor del contador de tiempo actual (temporizador) y lo compara con los valores de *var rangos_premios* para determinar el trofeo que ha ganado el jugador. Una vez sabido el trofeo se lo comunica al *Script* “Controlador de fase” mediante los parámetros *Var trofeo_bronce, plata* y *oro*

Asset tipo “Jugador” – Script “Controlador de juego”

Script general, adherido al *asset* tipo jugador, que confiere toda la funcionalidad necesaria para que el jugador pueda interactuar con los objetos que afectan a la jugabilidad de forma directa durante las fases de los mini-juegos. Concretamente permite que el jugador pueda interactuar con objetos como los juguetes del mini-juego “Recoger los juguetes” o el trapo del mini-juego “Limpiar los cristales”, mediante mecanismos *drag and drop*. Igualmente, mediante una variación (no contemplada en el diagrama de tareas) permite el manejo de la escoba en el mini-juego “Barrer el suelo”.

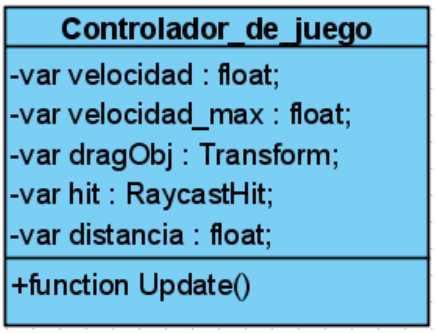


Figura 3.40: Diagrama del arquetipo de Script "Controlador de juego"

El funcionamiento del *Script* es sencillo. Mediante el método *Update()* se comprueba a cada momento si el usuario ha pulsado en la pantalla táctil. Si ese es el caso, el *Script* "lanza un rayo" (*var RaycastHit*) a los objetos de la escena hasta una distancia máxima (var *distancia*) a través del punto de la pantalla que ha pulsado el usuario. Si el rayo da a un objeto identificado como un objeto que afecta a la jugabilidad de forma directa (ej. Un juguete), se trata en el método como recogido. A partir de ahí, si el usuario mantiene el dedo pegado a la pantalla, el método moverá el objeto (modificará el componente *Transform* del objeto con *dragObj*) hacia donde el usuario desplace su dedo a una velocidad mínima de *var velocidad* y una máxima de var *velocidad_max*. De esta forma el jugador podrá controlar los objetos que afectan a la jugabilidad de forma directa a su gusto mediante mecanismos *drag and drop*.

Diagramas de relaciones

A continuación se presenta un diagrama con las relaciones entre los diferentes **arquetipos de *Scripts*** y los **tipos de *assets* de la tabla 3.4**, tratando a estos últimos como paquetes. Igualmente se muestra su relación con los **grupos de pantallas** identificados en el prototipo de bajo nivel de fidelidad, tratándolas como un paquete tipo modelo, en el diagrama.

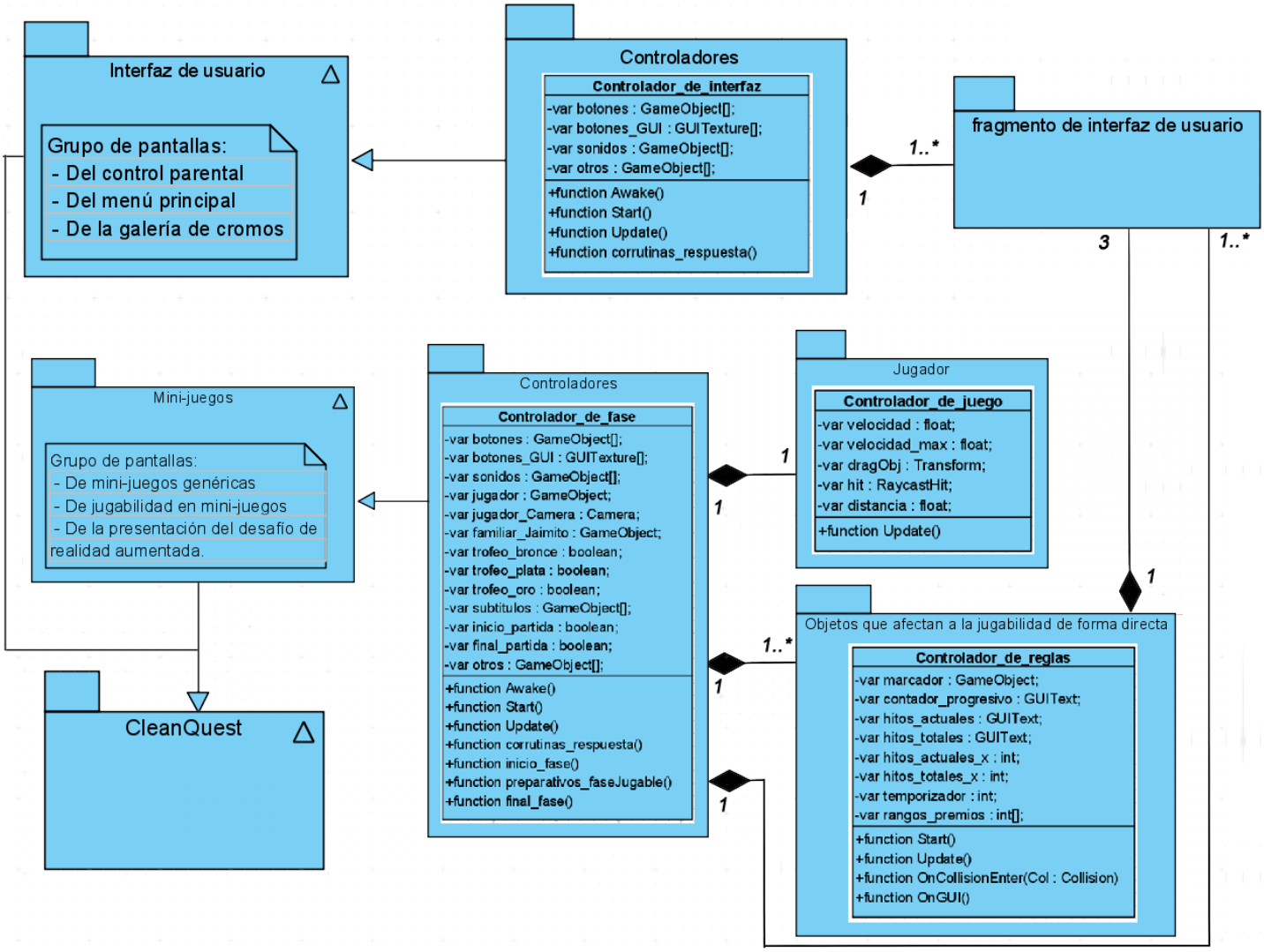


Figura 3.41: Diagrama de relaciones de Scripts arquetípicos en CleanQuest

3.4.3.3 Diagramas de secuencia

Con todos los arquetipos de *Scripts* establecidos y su relación como componentes dentro de los tipos de *assets* identificados (y con los grupos de pantallas), es posible elaborar un diagrama de secuencia. Este diagrama de secuencia muestra al **actor**, el jugador o usuario, y cada ***Script* arquetípico** perteneciente a una pantalla. El objetivo de estos diagramas es mostrar la sucesión de eventos en el tiempo, entre el actor y los *Scripts* y entre *Scripts*; ya sean las acciones que realiza el actor mediante la pantalla táctil, los métodos de los *Scripts* que se ejecutan y el *feedback* visual que devuelven al actor.

Para realizar el diagrama se ha decidido descartar las pantallas de los grupos de pantallas del control parental, del menú principal y de la galería de cromos. Estas pantallas son muy similares en cuanto a funcionamiento (con pequeñas variaciones específicas) y la forma que tienen de interactuar con el actor es demasiado simple, solamente respondiendo con corrutinas a las pulsaciones de los distintos botones

táctiles de la interfaz. En su lugar se ha optado por realizar un diagrama de secuencia de la combinación del resto de pantallas (exceptuando las pantallas de presentación del desafío de realidad aumentada), en otras palabras, **un diagrama de secuencia del transcurso de una fase típica en un mini-juego**, más concretamente y a modo de ejemplo, una fase del mini-juego "Recoger los juguetes". Se opta por este ejemplo debido a que se considera la interacción entre actor y *Scripts* más representativa y lo suficientemente compleja como para necesitar un diagrama de secuencia para ilustrar su flujo de eventos de cara a la producción.

La narrativa de eventos en el diagrama de secuencia, atendiendo a los eventos numerados que muestra, es la siguiente:

Antes de comenzar la fase, esto es, hacerse visible al usuario, el *Script* "Controlador de fase", prepara toda la escena para la *cutscene* de inicio de fase (1). A continuación le muestra la *cutscene* de inicio de fase (2,3 y 4) al usuario (la conversación inicial con el familiar de Jaimito). Asimismo, durante la conversación aparece en la interfaz el botón táctil "Empezar" (5) por si el usuario prefiere saltarse esta *cutscene*. El *Script* se queda esperando (6) hasta que el usuario pulse el botón, y cuando lo hace (7), el *Script* cambia la cámara del jugador a la necesaria para la fase jugable (11 y 12), comenzando de esta manera la ejecución del *Script* "Controlador de juego"(8). Igualmente activa el baúl de los juguetes, comenzando de esta manera la ejecución del *Script* "Controlador de reglas"(9). Así comienza la fase jugable y el *Script* "Controlador de fase" se queda esperando a que el usuario la complete la fase.

Cuando comienza la fase jugable, primeramente el *Script* "Controlador de reglas" (10) muestra en la interfaz de usuario el contador progresivo de tiempo y el contador de hitos o acciones numerables (13). Durante toda la fase jugable, el mismo irá incrementando el contador de tiempo progresivo cada segundo (14, 15 y 16). De esta forma el jugador ya puede concentrarse en jugar la fase jugable a través del *Script* "Controlador de juego".

Para completar la fase jugable, el jugador recoge cada juguete del escenario, mediante mecanismos *drag an drop* (12), y los introduce en el baúl de los juguetes (17 y 18). Cada vez que el jugador introduce un juguete en el baúl el *Script* "Controlador de reglas" lo detecta (19) e incrementa el contador de hitos (20 y 21). Cuando el mismo *Script* detecta que el jugador ha introducido el último juguete en el baúl (22, 23 y 24), se lo comunica al *Script* "Controlador de fase", indicándole el trofeo obtenido por el jugador en la fase (de acuerdo al tiempo empleado en completarla) terminando de esta manera la fase jugable (25) y pasando al final de la fase (26).

En el final de la fase se muestra al jugador la *cutscene* de final de fase (27), donde el familiar de Jaimito le presenta cada premio obtenido al completarla (31). Para avanzar de un premio a otro y para dar por finalizada la fase, el *Script* "Controlador de fase", muestra los botones táctiles correspondientes en la interfaz ("Siguiente" y "Terminar"), espera a su pulsación por parte del usuario y responde con corrutinas que contienen la funcionalidad de cada botón (28, 29, 30, 31, 32, 33, 34 y 35).

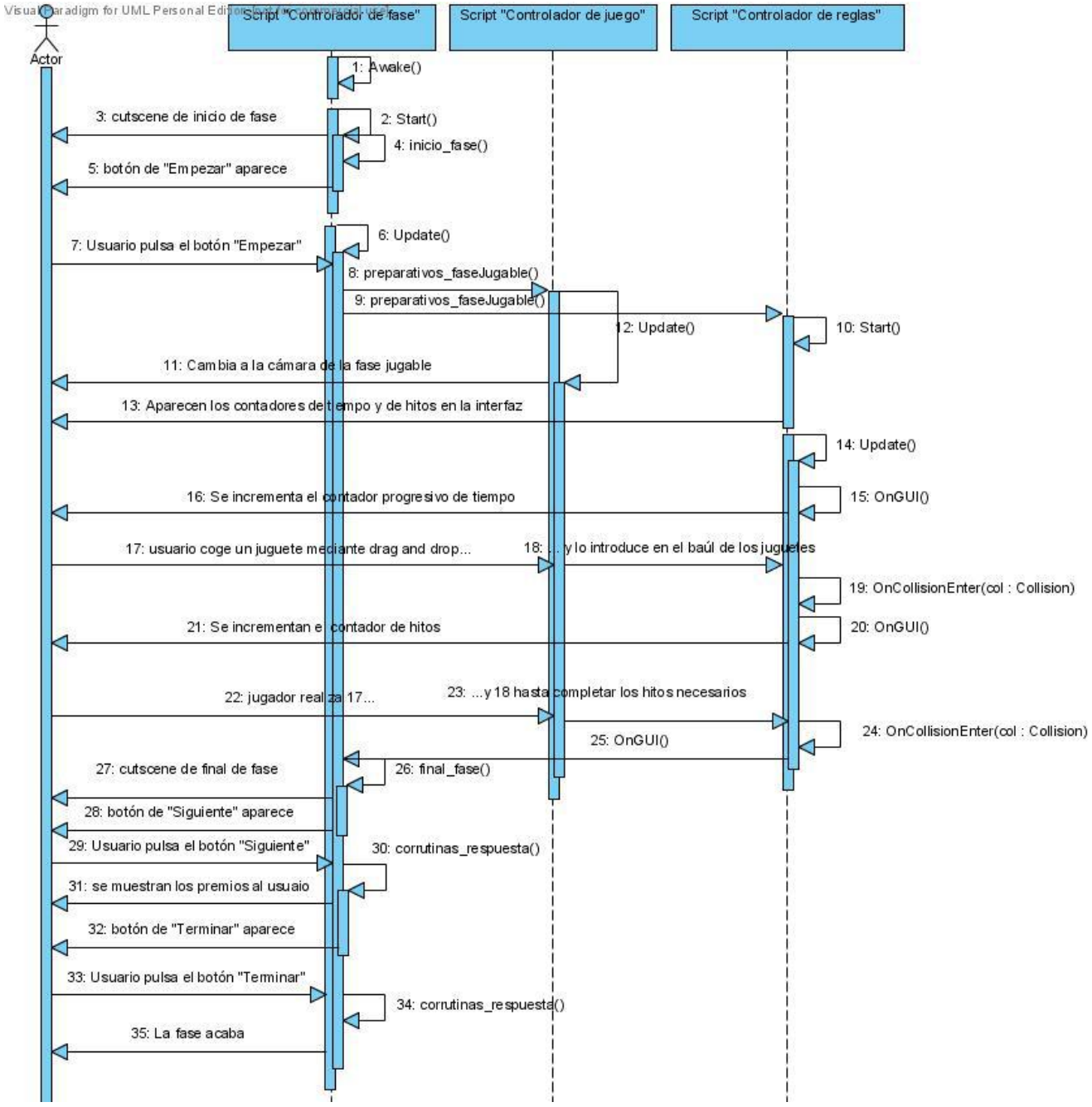


Figura 3.42: Diagrama de secuencia de una fase del mini-juego "Recoger los Juguetes"

3.5 Preproducción – Alternativas de diseño

Durante las sub-fases de concepción y el diseño del juego en la preproducción se consideraron **vías alternativas en diversos aspectos del diseño** de *CleanQuest*. Estas vías alternativas de diseño se descartaron en un primer momento debido a que se optó por la que se consideró como la vía óptima para resolver el problema planteado y por lo tanto la única reflejada en los correspondientes apartados de la memoria. En este apartado se comentan brevemente las vías alternativas de diseño más relevantes que se consideraron durante la preproducción, junto con su justificación para el descarte y clasificadas en cuanto a los aspectos del diseño que atañen.

El objetivo de este apartado es **recoger y documentar todas estas vías alternativas de diseño de cara a las fases de producción y pruebas**. Esto debido a que el aspecto global de la jugabilidad y en menor medida el diseño de *assets*, sigue una metodología de desarrollo cíclica (iterativa), y si en esas fases surgen problemas críticos con los mismos o el rechazo por parte de los usuarios que realizan las pruebas, se puede optar por elegir una de las vías alternativas documentadas como solución rápida que se adecúe más a la solución del problema.

Las vías alternativas de diseño comentadas se clasifican, según los distintos aspectos particulares expuestos en las sub-fases de preproducción anteriores: análisis del estado del arte, concepción, diseño conceptual y artístico y diseño de implementación y técnico.

3.5.1 Vías alternativas en el análisis del estado del arte

Todas las posibles vías alternativas en el estado del arte, ya sea respecto al uso de características de otros productos de software que comparten objetivos de la solución al problema propuesto o respecto al sistema y kit de desarrollo utilizados para el desarrollo de *CleanQuest* ya han sido cubiertas y justificadas en el propio apartado original.

3.5.2 Vías alternativas en la concepción

- **Jugador como Jaimito.** Se eligió esta opción, un avatar con nombre propio y ya integrado en el trasfondo, en lugar de un avatar genérico o personalizable, debido a que enriquece el trasfondo de forma directa (como si fuese una historia o relato a ojos del jugador) y facilita en gran medida el desarrollo. Aun así todavía se considera la posibilidad de que el avatar pueda ser personalizable por el jugador (ponerle su propio nombre, elegir sus características visuales, etc.).

- **Cromos coleccionables virtuales como incentivos.** Se eligió esta alternativa, como se explicó en el apartado original, inspirándose en el éxito de los monstruos virtuales coleccionables de la plataforma *Choremonster* y de franquicias similares como *Pokemon.* Además de que resultaba una opción más directa y sencilla de cara al desarrollo. Igualmente se consideraron otras alternativas similares a estos

premios dentro del juego, como dar puntos de forma intermedia que pudieran ser canjeados por los cromos (aprovechando el éxito del concepto de "dinero virtual"[32]) o emplear monstruos coleccionables como en *Choremonster* pero sustituyéndolos por animales de granja. Estas alternativas se consideraron demasiado costosas en cuanto al desarrollo, teniendo prácticamente el mismo efecto en el jugador que la escogida, así que se descartaron.

- **Tareas representativas para mini-juegos.** Como ya se comentó en el apartado original se eligieron las tareas de "Recoger los juguetes", "Barrer el suelo" y "Limpiar los cristales" basándose en las tareas de limpieza más representativas que se suelen asignar a los niños del rango de edad objetivo (6 a 9 años). También se consideraron otras tareas de limpieza, como "Recoger la mesa" o "Colgar la ropa", pero debido a su complejidad de cara al desarrollo (en cuanto a mecanismos de implementación) comparado con las elegidas, se descartaron. Aun así todavía se consideran para ampliar el juego con contenido de pago futuro.

3.5.3 Vías alternativas en el diseño conceptual y artístico

- **Diseño visual de elementos.** Por supuesto, para el diseño visual de los elementos del juego, incluyendo el diseño de personajes no jugables, de escenarios, objetos e interfaz, se consideraron y esbozaron otras alternativas. Se escogieron las mostradas en el apartado original ya que se pensó que eran los diseños visuales más adecuados atendiendo al estilo *cell shading* de los gráficos 3d y al planteamiento simple, intuitivo (reconocible) y desenfadado de la dirección artística en general. Como este aspecto es muy subjetivo de por sí, no se descartan del todo las alternativas.

- **Diseño funcional de elementos.** Este aspecto atañe sobre todo al diseño de los objetos que afectan a la jugabilidad de forma pasiva en los mini-juegos. Por ejemplo muebles, barreras invisibles o incluso el propio escenario que regulen la dificultad de una fase en un mini-juego de forma pasiva. Se han diseñado varios sets alternativos de estos objetos para cada fase en cada mini-juego. Cada set difiere solamente en la posición y/o tamaño de los objetos que contiene respecto a los demás. El set elegido para cada fase, no reflejado en esta memoria, ha sido escogido atendiendo a la dificultad teórica que debería tener cada fase. Aun así este es el aspecto que más posibilidades tiene de ser modificado durante las pruebas internas en la producción, debido a su facilidad para tolerar los cambios (ej. Simple cambio de posición o tamaño del modelo 3d de una silla).

3.5.4 Vías alternativas en el diseño de implementación y técnico

- ***Cutscenes* de inicio y final de fase.** Durante el diseño de *assets* en el apartado original, al comprobar la complejidad real de la integración de los personajes no

[32] Fuente: <http://en.wikipedia.org/wiki/Virtual_economy>.

jugables en *Unity*, en lo que respecta a la compatibilidad de formatos de modelo 3d, animaciones, etc. realizados con programas externos, contando además con poca experiencia inicial sobre el tema, se propuso alternativamente pre-renderizar estas *cutscenes*. Esto es, realizarlas enteramente en el programa externo blender, exportarlas en un formato de video compatible con *Unity* (.mp4) y reproducirlas directamente en *Unity*, sin la necesidad de una codificación substancial. Esta idea se descartó y en su lugar se hizo el esfuerzo por integrar los modelos 3d e animaciones de los personajes en *Unity*, ya que, aunque aumentaba la complejidad del desarrollo exponencialmente, era necesario para mantener el control sobre la cutscene, sobre todo al incluir elementos no prefijados como los trofeos y los cromos que obtiene el jugador al finalizar una fase en la *cutscene* de final de fase. Aun así la alternativa de los videos pre-renderizados se utilizó para realizar el video de introducción en el control parental y no se descarta del todo por si surgen problemas en la producción de estas cutscene.

- **Funcionalidad demasiado centralizada en *Scripts*.** Durante el diseño de *assets* y el diseño de la programación se decidió implementar la funcionalidad de todas las pantallas mediante unos *Scripts* arquetípicos, componentes de los tipos principales de *assets* establecidos. Estos *Scripts* arquetípicos son pequeños en número, relegando gran parte de la funcionalidad de una pantalla en un único *Script* usualmente componente del un *asset* de tipo “Controlador”, centralizando en él la funcionalidad de esa pantalla. Alternativamente se propuso un diseño más modular, aumentando el número de *Scripts*, siendo estos de menor peso (en código) y situándolos como componentes de cada *asset* que necesite funcionalidad. Por ejemplo, las animaciones y reproducción de sonido de un familiar de Jaimito en la *cutscene* inicial de una fase estarían controladas por un pequeño *Script* adherido al *asset* que represente al personaje, en lugar de por el arquetipo de *Script* “Controlador de fase” centralizado que finalmente se decidió utilizar. Esta alternativa se descartó, ya que es muy costoso controlar el funcionamiento de tantos *Scripts* en armonía, sobre todo estando adheridos a *assets* inestables en la escena, que pueden activarse o desactivarse varias veces según lo requiera el curso de eventos.

3.6 Preproducción – Planificación

La fase de planificación es la siguiente sub-fase después del diseño y la última dentro de la pre-producción. En ella se **planifican todas las tareas de implementación a realizar en la fase de producción** respecto al tiempo disponible para el resto del desarrollo. El objetivo de esta sub-fase dentro de la preproducción es gestionar el tiempo restante del desarrollo para la fase de producción, de la forma más óptima posible.

La planificación de estas tareas se representa gráficamente utilizando un **diagrama Gantt simple**. En este diagrama se reflejan las tareas planificadas de la fase de producción y además se incluirán las tareas realizadas en la preproducción para mantener la coherencia en la gestión del tiempo. Para identificar las tareas necesarias dentro de la producción y estimar su tiempo de duración se atenderá principalmente a los requisitos de la especificación de requisitos (apartado 3.3) y al diseño de implementación y técnico en general (apartado 3.4). De forma secundaria se atenderá a las decisiones de diseño del diseño conceptual y artístico, sobre todo de cara a la elaboración de *assets* con programas externos, como modelos 3d o sonidos. Por supuesto, también se tendrán en cuenta todos los eventos u obligaciones personales que influyan en el desarrollo para la estimación de la duración en cada tarea.

A continuación se muestran los diagramas Gantt de la preproducción y de la posterior fase de producción, junto con algunas notas aclarativas sobre eventos o situaciones relevantes para el proceso de desarrollo:

- **Vacaciones en agosto (Agosto: Vacaciones).** En la tutoría que se llevó a cabo el 29 de agosto se determinó que iba a ser prácticamente imposible acabar todo el desarrollo antes de la fecha indicada para la última sesión del TFG en el curso 2013 – 2014. Así que se decidió entregarlo en la próxima convocatoria, primera sesión del curso 2014 – 2015 y se reservó agosto como mes de vacaciones.

- **Redacción de la memoria.** La memoria del proceso de desarrollo y del TFG en general se va redactando a la par que la realización de cada fase del proceso. En la duración de cada tarea se incluye el proceso de redacción de la parte correspondiente en la memoria. En el diagrama siguiente, las tareas reservadas para la memoria (Enero: memoria) están pensadas para redactar fragmentos teóricos de la misma no ligados al proceso de desarrollo real en este TFG (como la Postproducción) y para formalizar y maquetar todos los apartados.

- **Pruebas de sistema (Diciembre: Producción parte 3).** Las pruebas de sistema centradas en usuarios (pruebas con niños del rango de edad objetivo) se realizarán en el periodo de vacaciones navideñas, aprovechando la reunión familiar con parientes con el rango de edad objetivo.

Nombre	Fecha ...	Fecha de fin
Julio SEM 1: Definición del problema	2/07/14	6/07/14
Reunión Tutor	2/07/14	2/07/14
Exposición de problema, solución ...	3/07/14	6/07/14
Julio SEM 2 y 3: Estado del arte	7/07/14	20/07/14
Investigación de soluciones del pr...	7/07/14	8/07/14
Análisis de sistema y kits de desar...	9/07/14	14/07/14
Análisis de productos de SW	9/07/14	14/07/14
Conclusiones del análisis	15/07/14	20/07/14
Julio SEM 4: Concepción parte 1	21/07/14	28/07/14
Descripción del juego	21/07/14	24/07/14
Selección de tareas de limpieza	25/07/14	28/07/14
Agosto: Vacaciones*	29/07/14	31/08/14
Reunión Tutor	29/07/14	29/07/14
Vacaciones	29/07/14	31/08/14
Septiembre SEM 1 y 2: Concepción pa...	1/09/14	14/09/14
Descripción de la jugabilidad	1/09/14	4/09/14
Dirección artística y trasfondo	2/09/14	6/09/14
Descripción del control parental	2/09/14	10/09/14
Estructura y flujo del juego	7/09/14	10/09/14
Estudio de viabilidad de ideas	9/09/14	14/09/14
Septiembre SEM 3 y 4: Diseño concep...	15/09/14	28/09/14
Diseño de PNJ	15/09/14	21/09/14
Diseño de objetos y escenarios	15/09/14	21/09/14
Diseño de sonido	21/09/14	21/09/14
Diseño del guión	16/09/14	19/09/14
Diseño de la interfaz de usuario	22/09/14	28/09/14
Diseño del control parental, cromo...	24/09/14	28/09/14
Octubre SEM 1: Especificación de req...	29/09/14	5/10/14
Especificación de requisitos	29/09/14	5/10/14
Octubre SEM 2 y 3: Diseño de implem...	6/10/14	23/10/14
Prototipo de bajo nivel de fidelidad	6/10/14	10/10/14
Selección de herramientas secun...	6/10/14	6/10/14
Diseño de la programación	11/10/14	20/10/14
Planificación (Gantt)	21/10/14	23/10/14

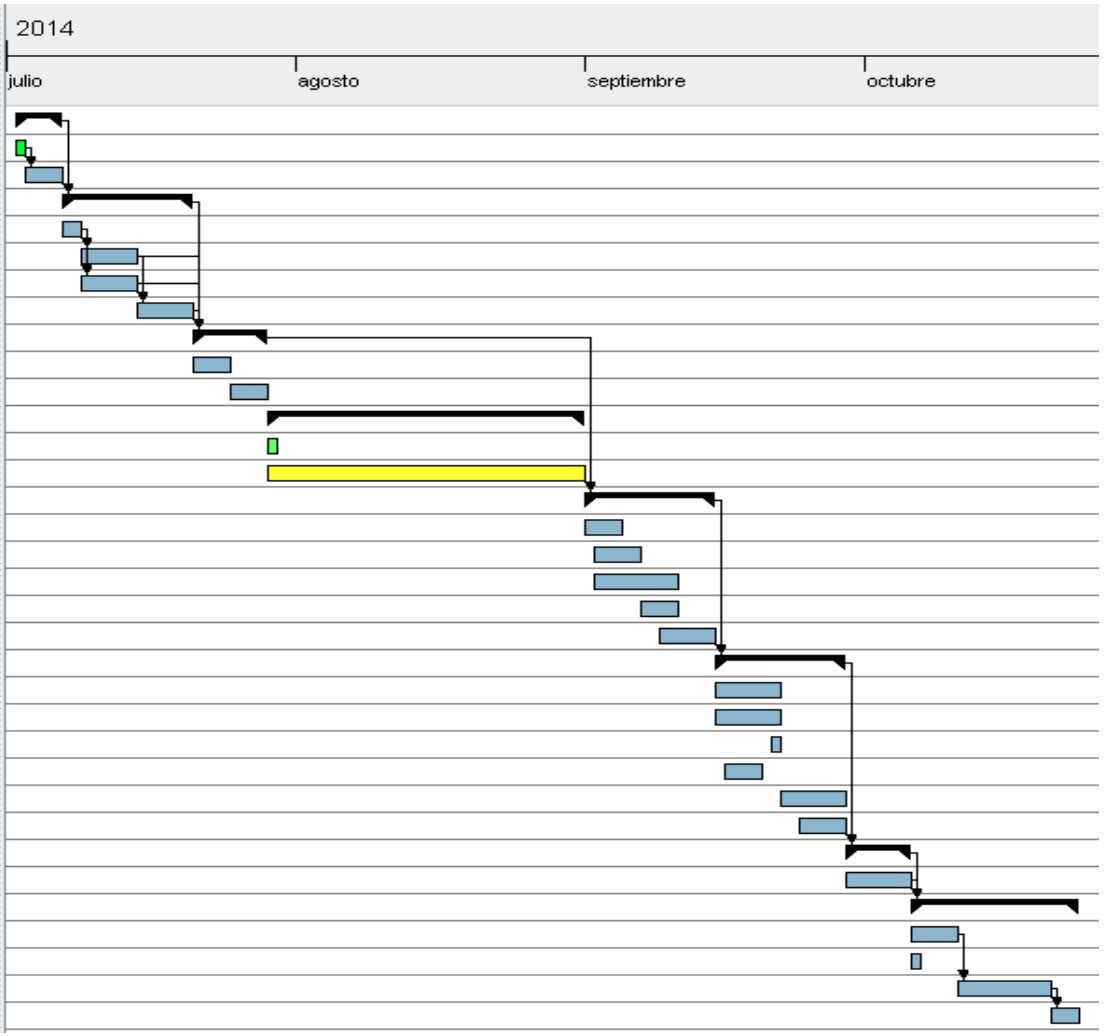
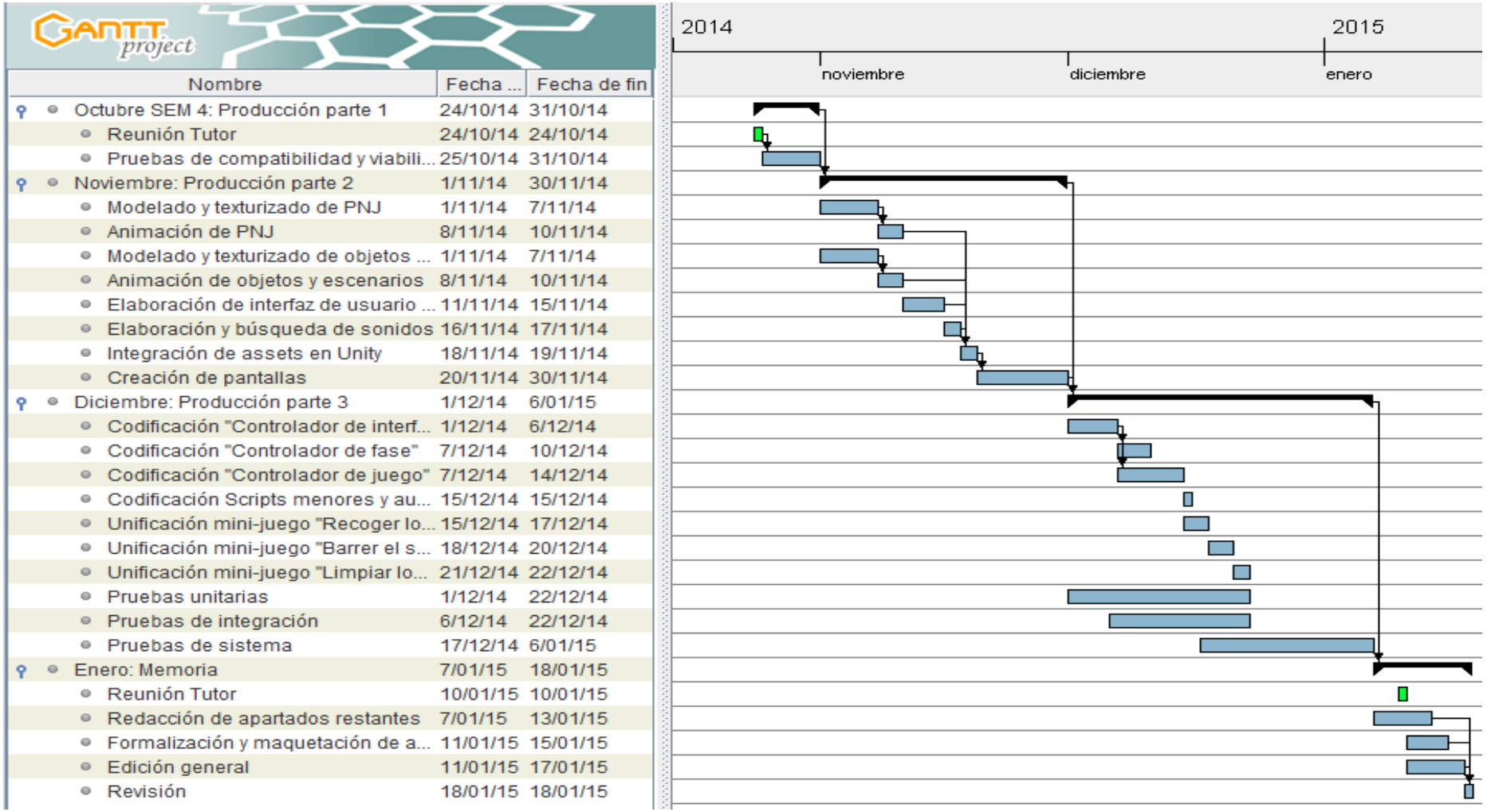


Figura 3.43: Diagrama Gantt de la fase de preproducción de CleanQuest



Nombre	Fecha ...	Fecha de fin
Octubre SEM 4: Producción parte 1	24/10/14	31/10/14
Reunión Tutor	24/10/14	24/10/14
Pruebas de compatibilidad y viabili...	25/10/14	31/10/14
Noviembre: Producción parte 2	1/11/14	30/11/14
Modelado y texturizado de PNJ	1/11/14	7/11/14
Animación de PNJ	8/11/14	10/11/14
Modelado y texturizado de objetos ...	1/11/14	7/11/14
Animación de objetos y escenarios	8/11/14	10/11/14
Elaboración de interfaz de usuario ...	11/11/14	15/11/14
Elaboración y búsqueda de sonidos	16/11/14	17/11/14
Integración de assets en Unity	18/11/14	19/11/14
Creación de pantallas	20/11/14	30/11/14
Diciembre: Producción parte 3	1/12/14	6/01/15
Codificación "Controlador de interf...	1/12/14	6/12/14
Codificación "Controlador de fase"	7/12/14	10/12/14
Codificación "Controlador de juego"	7/12/14	14/12/14
Codificación Scripts menores y au...	15/12/14	15/12/14
Unificación mini-juego "Recoger lo...	15/12/14	17/12/14
Unificación mini-juego "Barrer el s...	18/12/14	20/12/14
Unificación mini-juego "Limpiar lo...	21/12/14	22/12/14
Pruebas unitarias	1/12/14	22/12/14
Pruebas de integración	6/12/14	22/12/14
Pruebas de sistema	17/12/14	6/01/15
Enero: Memoria	7/01/15	18/01/15
Reunión Tutor	10/01/15	10/01/15
Redacción de apartados restantes	7/01/15	13/01/15
Formalización y maquetación de a...	11/01/15	15/01/15
Edición general	11/01/15	17/01/15
Revisión	18/01/15	18/01/15

Figura 3.44: Diagrama Gantt de la fase de producción de CleanQuest

3.7 Producción

La producción es la fase del proceso de desarrollo donde **se realiza toda la implementación necesaria de *CleanQuest***; no sólo codificación, sino también modelado 3d y 2d, elaboración de animaciones, etc. Esta fase continúa la fase de preproducción y se basa en todas las decisiones tomadas en ella para realizar la implementación, centrándose principalmente en la especificación de requisitos, el prototipo de bajo nivel de fidelidad y el diseño de implementación y técnico de la preproducción.

El objetivo de esta fase es desarrollar un **prototipo software de alto nivel de fidelidad** de *CleanQuest*, con un grado de funcionalidad y detalle suficiente para poder utilizarlo en la **fase de pruebas (apartado 4)**, en especial de cara a los usuarios que tomarán parte en ella.

En esta memoria **no se registra esta fase en detalle**, ya que el objetivo de la memoria es exponer el proceso de desarrollo de *CleanQuest* centrándose en las decisiones de preproducción tomadas, que es donde realmente se justifica y se da a entender más fácilmente el proceso en sí. En su lugar se expondrán brevemente los eventos relevantes acaecidos en la implementación, así como cambios importantes que hubieran podido surgir respecto a lo que se tenía planificado en la fase de preproducción.

3.7.1 Proceso de implementación

En este sub-apartado se muestran capturas de pantalla representativas del proceso de implementación relativo a la elaboración de los personajes no jugables, de los objetos y escenarios y de las pantallas del juego en general. Se ha elegido mostrar este tipo de aspectos ya que son los más visuales de cara al prototipo de software final, a diferencia del proceso de codificación. A continuación se muestra el proceso de implementación de algunos de los elementos más representativos mencionados, a modo de ejemplos, junto con su aspecto final en el prototipo de software de alto nivel de fidelidad:

- **Implementación del personaje no jugable, "Hermano de Jaimito".** De izquierda a derecha, arriba a abajo: boceto inicial, modelado y texturizado con el programa *Sculptris*, animado con el programa *Blender* y por último su integración final en *Unity*, en la pantalla *(Ej)JUG-GEN1* (atendiendo al prototipo de baja fidelidad, apartado 3.4.1), ya perteneciente al prototipo software de alto nivel de fidelidad. En esta pantalla se pueden observar igualmente otros elementos finales del mini-juego "Recoger los juguetes": una parte del escenario, algunos juguetes y la interfaz de la fase jugable.

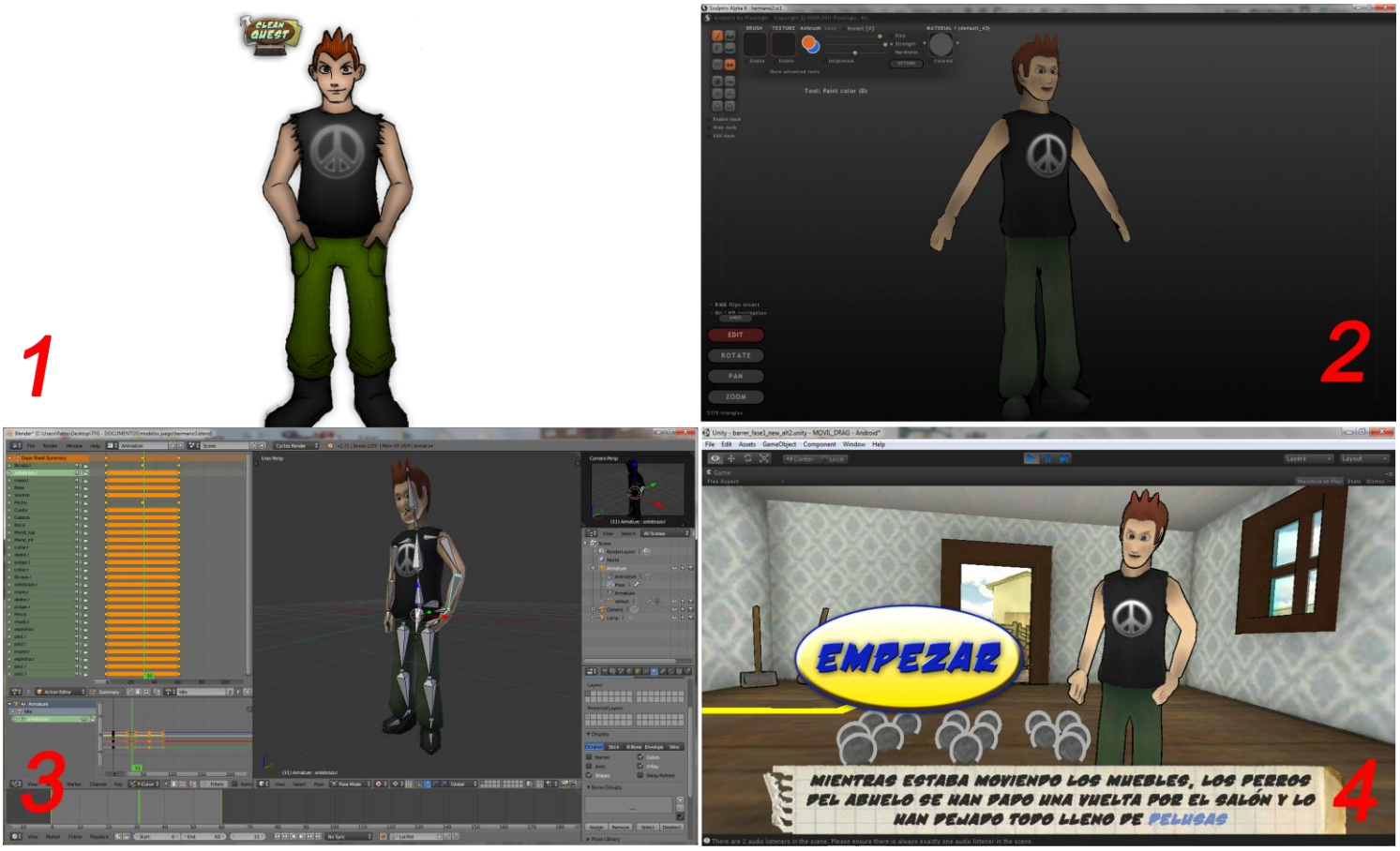


Figura 3.45: Proceso de implementación del PNJ "Hermano de Jaimito"

- **Implementación de objetos y escenarios, "Baúl de los juguetes".** De izquierda a derecha, arriba a abajo: boceto inicial, modelado y texturizado con el programa *Sculptris*, Animado con el programa *Blender* y por último su integración final en *Unity*, en la pantalla *(Ej)JUG-RE* (atendiendo al prototipo de baja fidelidad, apartado 3.4.1), ya perteneciente al prototipo software de alto nivel de fidelidad. En esta pantalla se pueden observar igualmente otros elementos finales del mini-juego "Recoger los juguetes": una parte del escenario, algunos juguetes y la interfaz de la fase jugable.

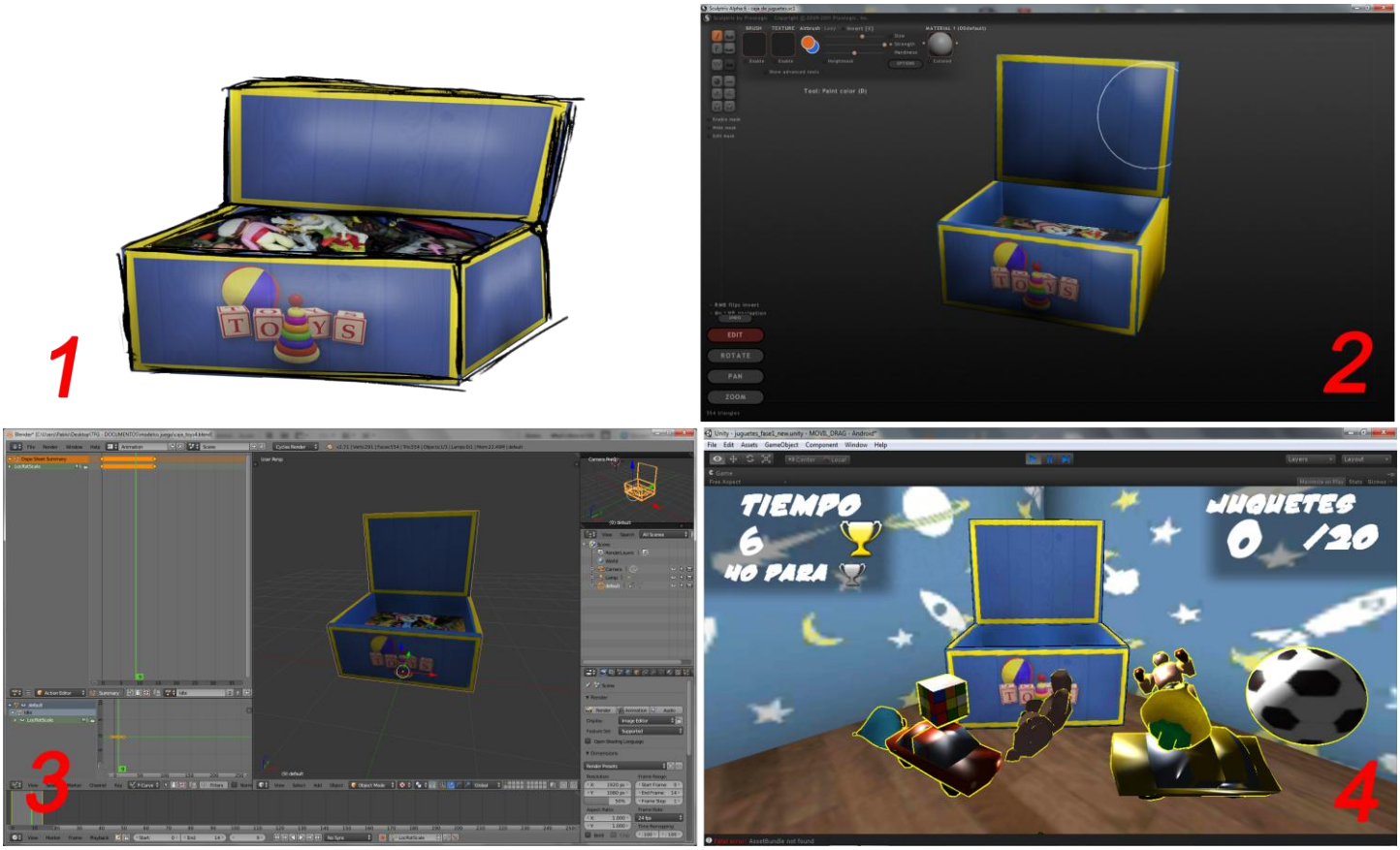


Figura 3.46: Proceso de implementación del objeto "Baúl de los juguetes"

- **Implementación de la pantalla, "Menú principal" (MP).** De izquierda a derecha, arriba a abajo: *mockup* inicial y resultado final en *Unity*. Nótese como se han aprovechado los elementos y la maquetación de la interfaz del *mockup* directamente en la creación de la pantalla en *Unity.*

Figura 3.47: Proceso de implementación del la pantalla "Menú principal" (MP)

3.7.2 Cambios significantes respecto a la preproducción

En este sub-apartado se comentan de forma breve los cambios más críticos o relevantes sucedidos durante la fase de producción, respecto a lo que se tenía planeado en la fase de preproducción. Estos cambios han podido producirse por dificultades en el proceso de implementación no previstas en la preproducción, decisiones estéticas o superficiales de última hora o resultado de la fase de pruebas.

A continuación se exponen estos cambios justificados, separados por puntos:

- **Diversos cambios estéticos y superficiales en objetos y escenarios.** Tanto el aspecto visual como el posicionamiento y tamaño en la escena de varios objetos que afectan a la jugabilidad de forma pasiva y de los escenarios, han sido ligeramente modificados (respecto a su estado inicial), a la hora de integrarlos en *Unity*. El motivo ha sido la necesidad de regular la dificultad inicial de ciertas fases de los mini-juegos. Este cambio en concreto ya se preveía desde la preproducción, ya que la única forma de regular realmente la dificultad en una fase es mediante ensayo y error, realizando pruebas en las fases jugables paralelamente en la producción.

- **Clips de música de músicas con derechos de autor.** Como se comentó en el apartado de "Diseño conceptual y artístico", en un principio, las pistas de música para cada pantalla se seleccionarían de forma externa (ya que no se posee ninguna experiencia en composición de música) dando prioridad a los repositorios gratuitos sin derechos de autor en internet. El problema es que durante la producción, a la hora de buscarlas en estos repositorios, no se encontraron pistas los suficientemente aptas para representar cada pantalla y la calidad del juego en general. Por este motivo se ha decidido utilizar pistas de músicas comerciales (con derechos de autor) de forma provisional para el prototipo software de alto nivel.

- **Fases del mini-juego "Recoger los juguetes" en diferentes perspectivas.** En la preproducción se decidió plantear cada fase del mini-juego "Recoger los juguetes" con una única perspectiva en primera persona, fija, para el jugador. Esto es, el jugador vería todos los juguetes repartidos por la habitación y el baúl de los juguetes dentro de una única pantalla en primera persona. Durante la producción se comprobó que una única pantalla no permitía incluir muchos juguetes en pantalla debido a las limitaciones de espacio físico de la pantalla y porque si se intentaba alejar la cámara (para incluir más espacio), los juguetes se veían demasiado pequeños como para ser controlados de forma cómoda por el jugador. Por este motivo se decidió partir cada fase del mini-juego en diversas perspectivas dentro de la misma escena, siempre en primera persona, en distintas partes del escenario (la habitación de Jaimito). Esto es, dentro de una misma fase, cuando el jugador recoja los juguetes en una parte de la habitación, la cámara del jugador junto con el baúl de los juguetes cambiará hacia otra parte de la habitación con más juguetes para que los recoja el jugador. Este cambio de cara a la codificación es mínimo, ya que solamente conlleva un cambio de posicionamiento de la cámara del jugador cuando un número determinado de muñecos han sido recogidos en la fase.

3.8 Postproducción

La fase de Postproducción es la siguiente y última fase del proceso de desarrollo, una vez finalizada la producción. Esta fase se perpetúa durante todo el ciclo de vida comercial restante del videojuego. En ella se realiza el mantenimiento del videojuego una vez finalizado el mismo, comenzando por su despliegue comercial y seguido por su posterior gestión de expansiones de contenido, parches o actualizaciones que añadan funcionalidades o corrijan errores.

Esta fase **no será realizada realmente y sólo será documentada en la memoria de forma teórica**, esto es, estableciendo un **modelo de negocio definitivo** y un **plan de gestión de parches y actualizaciones** futuras para *CleanQuest*. Esto es debido a que no se pretende y no hay tiempo suficiente en la realización de este TFG para desplegar el producto finalizado en plataformas comerciales y estudiar su comportamiento en las mismas. Igualmente, como se comentó en el apartado de "Producción" (apartado 3.7), el objetivo de esta memoria es exponer el proceso de desarrollo de *CleanQuest* centrándose en las decisiones de preproducción tomadas, terminando de esta manera el proceso de desarrollo real con las pruebas de validación de producto (pruebas con usuarios), utilizando el prototipo de software de alto nivel de fidelidad.

Para establecer el modelo de negocio definitivo (plan de despliegue) y el plan de parches y actualizaciones de *CleanQuest* hay que atender, por un lado, al **modelo de negocio base** esbozado en el sub-apartado "Modelo de negocio base" (apartado 2.2.7) de las conclusiones del análisis del estado del arte, y por otro, a las características finales del **prototipo software de alto nivel de fidelidad** de *CleanQuest*, resultado de la fase de producción.

3.8.1 Modelo de negocio definitivo

El modelo de negocio definitivo de *CleanQuest* hace referencia al tipo de despliegue que se hará con la versión definitiva del mismo. En otras palabras, en que plataforma comercial es más viable y óptimo desplegarlo y la manera de monetizarlo para que pueda cumplir su función de forma correcta (herramienta para solucionar el problema de la desobediencia o indiferencia infantil ante la colaboración en las tareas de limpieza del hogar) y además aporte beneficios.

En el sub-apartado "Modelo de negocio base" (apartado 2.2.7) de las conclusiones del análisis del estado del arte, se esbozó un posible modelo de negocio base viable para *CleanQuest*, justificando su uso. Para ello se atendió a los modelos de negocio de los productos de software analizados y a las características base iniciales establecidas para *CleanQuest*. El modelo de negocio en cuestión es **distribución gratuita con la inclusión de micro pagos opcionales futuros**, esto es, los usuarios pueden descargarse el juego base (todo lo incluido en el prototipo software de alto nivel de fidelidad) de forma gratuita y tienen la posibilidad de descargar contenido adicional que expanda el juego, por una pequeña cantidad de dinero. En el mismo apartado se indicó que hasta no conocer las características finales del prototipo e

software de alto nivel de fidelidad de *CleanQuest* este modelo de negocio base seguiría siendo provisional.

Ya finalizadas la preproducción y la producción (incluida la fase de pruebas), con el prototipo de software presente, es posible confirmar y especificar más este modelo de negocio base para *CleanQuest*, estableciendo un modelo de negocio definitivo.

El prototipo software de alto nivel de fidelidad, después de todas las pruebas y modificaciones necesarias, presenta unas características finales con apenas cambios respecto a las características base iniciales (premisa, dirección artística, jugabilidad, trasfondo y género) establecidas en el análisis del estado del arte. Por este motivo se ha decidido continuar con el mismo modelo de negocio que se esbozó en el análisis del estado del arte.

Continuando con el mismo modelo de negocio inicial, es necesario especificarlo, seleccionando la **plataforma comercial** actual más adecuada para su distribución gratuita y respecto al aspecto de los **micro pagos**, concretar que contenido del juego sería el más adecuado ofertar (expandir) a través de los mismos para garantizar su sostenibilidad.

Plataforma comercial de distribución

Para la distribución gratuita de *CleanQuest*, considerando su condición como aplicación para dispositivos móviles con sistema operativo *Android*, la mejor opción para su distribución es la propia plataforma de distribución digital oficial del propietario de *Android*, ***Google Play Store*** [48]. Esta plataforma se centra en la distribución de aplicaciones *Android* y es la más popular y cercana a los usuarios con este tipo de dispositivos. La búsqueda de aplicaciones en el mismo es intuitiva, teniendo un motor de búsqueda potente que clasifica a las aplicaciones según criterios globales; lo que ayudaría en gran medida para presentar a *CleanQuest* a los usuarios como un videojuego de carácter educativo.

De cara al desarrollador, es una plataforma abierta (sin apenas poner trabas para la publicación de aplicaciones), fácil de utilizar y económica, solamente teniendo que pagar 25$ para poder publicar cualquier número de aplicaciones en ella. Además, incluye un sistema de seguimiento para comprobar cómo se comporta la aplicación publicada en la comunidad.

Contenido de los micro pagos

Como ya se comentó durante el apartado del análisis del estado del arte (apartado 2), una de las razones por las que se consideró la adición de contenido extra en *CleanQuest* mediante micro transacciones o micro pagos (una vez adquirido el juego base de forma gratuita), es debido al carácter modular de *CleanQuest*. Sobre todo al tratar el juego como un conjunto de mini-juegos, siendo estos a su vez un conjunto de fases, cuyos premios son cromos coleccionables. De esta forma se podría considerar el contenido de estos micro pagos como **nuevos cromos virtuales coleccionables que se añadan a la colección, fases extra para mini-juegos existentes o incluso**

mini-juegos completamente nuevos (que representen otras tareas de limpieza), todo ello por supuesto de forma opcional.

De igual forma, la elección de la plataforma *Google Play Store* como plataforma de distribución para *CleanQuest* es ideal para este tipo de actualizaciones de pago, ya que ofrece un sistema adaptado para este tipo de micro transacciones que facilita al desarrollador su distribución.

3.8.2 Gestión de parches y actualizaciones

Los parches y actualizaciones de *CleanQuest* son la adición de contenido o funcionalidades que optimicen o corrijan contenido o funcionalidades ya existentes en el mismo. Su distribución se realiza por la misma plataforma que los miccro transacciones (*Google Play Store*), pero a diferencia de ellas, son gratuitas, ya que forman parte del compromiso con el usuario que descarga el juego.

No es posible especificar un plan de parches y actualizaciones concretos en esta memoria ya que la única forma de conocer su contenido o funcionalidad añadida es cuando verdaderamente exista su necesidad. Para conocer el momento de necesidad de estos parches y actualizaciones se realizarán pruebas con usuarios y revisiones periódicas del videojuego para encontrar puntos del mismo que sea necesario optimizar o corregir. Igualmente, para identificar más puntos del videojuego necesitados de parches o actualizaciones, se atenderá al *feedback* de los usuarios que hayan adquirido el producto de forma ordinaria, a través del tablón de opiniones de los usuarios que aporta la propia *Google Play Store* o mediante el contacto con los mismos a través de futuras cuentas de *Twitter* o *Facebook*.

4.Pruebas

La fase de pruebas es una sub-fase de la producción dentro del desarrollo. Ha sido relegada a este apartado dedicado debido a su relevancia de cara al proceso de desarrollo, ya que es la parte de todo el proceso que realmente demuestra la validez del producto para cumplir su propósito, y para facilitar su comprensión lectora.

Esta fase tiene lugar de forma paralela a la fase de producción (y en parte a la postproducción). En ella se comprueba el funcionamiento de todos los módulos de *CleanQuest* que van siendo implementados y una vez terminada la implementación, ya con el prototipo de software de alta fidelidad finalizado, se realizan las pruebas de sistema y de validación de producto (pruebas con usuarios) con el mismo. Todas estas pruebas **se centran en aspectos relativos a la jugabilidad** en mini-juegos o del control del juego en general (interfaces) por parte del usuario. Esto es debido a que, como ya se indicó en el sub-apartado de "Metodología y medios empleados" de la introducción, sólo se utilizan metodologías iterativas en aspectos de este tipo (aspectos de la jugabilidad); por lo que si las pruebas en estos aspectos resultan negativas, **el desarrollo puede soportar una nueva iteración para ellos**.

Las pruebas realizadas se dividen en 4 tipos de pruebas: **pruebas unitarias** (fragmentos o módulos individuales del juego), **pruebas de integración** (conjunto de pruebas unitarias) y **pruebas de sistema**, y de **validación del producto**. Cabe destacar que, debida al alto número de pruebas realizado durante todo el proceso de implementación, sólo se reflejarán en esta memoria las pruebas realizadas más representativas de cada tipo a modo de ejemplo.

4.1 Pruebas unitarias

Las pruebas unitarias son aquellas cuyo objetivo es comprobar y validar la funcionalidad de fragmentos de código o métodos concretos de un producto de software. Debido a su naturaleza, estas pruebas usualmente son realizadas de forma paralela a la fase de producción, cada vez que se implementa algún módulo o fragmento individual del videojuego, para validar su funcionamiento.

Como ya se comentó en el apartado “Diseño de la programación” (apartado 3.4.3) en la preproducción, al utilizar *Unity*, la implementación de *CleanQuest* no se basa únicamente en la codificación de *Scripts*, sino en la combinación de *Scripts* con la integración correcta de los *assets* que forman una escena en el editor (*framework*) de *Unity*. Por este motivo, para este tipo de pruebas (y para el resto), es necesario incluir y tener en consideración tanto los fragmentos de *Scripts* como los *assets* particulares que actúen de forma conjunta a ese fragmento, de forma similar a los diagramas de clases (apartado 3.4.3.2) en la preproducción.

Atendiendo a estas nociones, para realizar las pruebas unitarias, en vez de utilizar simplemente métodos de un *Script* con una entrada y salida de parámetros (forma convencional para las pruebas unitarias) esperadas, se utilizarán “**Acciones**”. Estas “Acciones” serán consideradas como un conjunto de al menos uno de los siguientes elementos:

- Uno o varios métodos de un *Script*, independientemente de si cuentan con parámetros de entrada o salida.
- Uno o varios *assets*, junto con sus correspondientes componentes en la escena.
- Una o varias interacciones del usuario con el dispositivo que desencadene o forme parte del proceso.

Todos los elementos que formen una “Acción”, actuarán de forma conjunta a modo de proceso, con una entrada determinada y una salida esperada. La **entrada** puede ser: Cualquier interacción iniciada por el usuario en el dispositivo; cualquier *asset*, interacción de *asset* o componente de un *asset* desde el punto de vista del editor; cualquier parámetro o ejecución de un método; o un conjunto de los anteriores. La **salida**, por su parte, puede ser: Algún tipo de *feedback* (visual, sonoro, etc.) del videojuego hacia el usuario; Un nuevo *asset*, componente de *asset* o una modificación de los ya existentes en la escena; cualquier parámetro o ejecución de un método; o un conjunto de los anteriores.

Teniendo en cuenta todo esto, las pruebas unitarias realizadas se dividen en dos tipos: **Pruebas unitarias de caja negra** y **pruebas unitarias de caja blanca**.

4.1.1 Pruebas unitarias de caja negra

Las pruebas unitarias de tipo caja negra realizadas, son aquellas en las que se comprueba si al ejecutar una "Acción" con una entrada determinada, resulta en una salida esperada, sin tener en cuenta el proceso interno por el que la "Acción" ha convertido esa entrada en esa salida.

Como se ha comentado anteriormente, las pruebas documentadas en esta memoria son las más representativas de cada tipo. En este apartado sólo se registran las pruebas unitarias de caja negra de "Acciones" ejemplares; esto es, "Acciones" que al ser validadas a través de estas pruebas, puedan validar otras "Acciones" similares.

Para documentar estas pruebas se utiliza una tabla con la siguiente información en cada una de sus columnas:

- **Acción.** Nombre o descripción corta identificativa de la "Acción".
- **Pantalla.** Pantalla donde sucede la "Acción". Se utilizarán los identificadores de pantalla utilizados en el prototipo de bajo nivel de fidelidad, ya que coinciden con las pantallas del prototipo software de alto nivel de fidelidad fruto de la producción.
- **Entrada.** Listado de los elementos dentro del conjunto de entrada.
- **Salida esperada.** Listado de los elementos dentro del conjunto de salida esperados.
- **Salida real.** Listado de los elementos dentro del conjunto de salida realmente devueltos. Puede incluir el número de intento para esa prueba si procede.

Acción	Pantalla	Entrada	Salida esperada	Salida real
Apertura de teclado virtual táctil	(Ej)(Sub)CP-ICL (Control parental)	*Usuario pulsa botón "Entrada de texto"; componente GUITexture del asset boton_codigo_RJuguetes; ejecución método OnGUI() del controlador de interfaz.*	*Apertura del teclado táctil nativo de Androide en primer plano*	La esperada.
Introducción y reconocimiento de texto en el teclado virtual táctil.	(Ej)(Sub)CP-ICL (Control parental)	*Usuario introduce texto a través del teclado virtual; ejecución método OnGUI() del controlador de interfaz.*	*Cierre del teclado táctil nativo de Androide en primer plano. Parámetro String código_introducido que contiene la cadena de texto introducido en el teclado.*	La esperada.
Gestión de *arrays* de *assets* (visor de cromos)	(Sub)GC-VC (Visor de cromos)	*Usuario pulsa un cualquier cromo en el visor de cromos; componente GUITexture del asset pack_cromos[]; ejecución método Update()*	*Ejecución de método mostrar_propiedades() del controlador de interfaz. Se muestra el modelo del cromo*	La esperada

		del controlador de interfaz.	*rotatorio y su nombre en pantalla.*	
Gestión de *cutscene* (válido para inicio y final de fase)	*(Ej)JUG-GEN1*	*Ejecución de la corrutina inicio_fase() del controlador de fase; asset familiar_Jaimito con su componente Animation; asset pack_voz_familiar[] con su componente Audio Source; Asset Player con su componente Camera.*	*Muestra de la cutscene de forma correcta, esto es, desde el ángulo de cámara correcto, el familiar de Jaimito animándose de forma correcta y los clips de voz correspondientes reproduciéndose en orden.*	La esperada.
Prueba mecanismo *drag and drop*	(Ej)JUG-RE (Fase jugable del mini-juego "Recoger los juguetes")	*Usuario pulsa y mantiene uno de los juguetes de la pantalla táctil. Lo arrastra por ella y lo suelta; ejecución del método Update() del controlador de juego; asset juguetes[].*	*El juguete se mueve por la pantalla acompañando el dedo mantenido del usuario. Al soltarlo el juguete deja de seguirlo y cae de una forma coherente.*	1ª Prueba: Los juguetes no responden a la pulsación del usuario. Resultaba que la variable que representaba la distancia de alcance para coger los juguetes no era suficiente. Se vuelve a calibrar. 2ª Prueba: La esperada.
Prueba reconocimiento de colisiones	(Ej)JUG-RE (Fase jugable del mini-juego "Recoger los juguetes")	*Usuario suelta un juguete que va a parar al espacio que ocupa el baúl de los juguetes en la escena; ejecución de la corrutina OnCollisionEnter() del controlador de reglas; asset baul_juguetes con su componente Collider*	*El asset juguete arrojado se destruye en cuanto toca ese espacio; el parámetro entero jug_cont del controlador de reglas se incrementa en uno.*	La esperada.
Prueba control de escoba mediante Joystick táctil	(Ej)JUG-BA (Fase jugable del mini-juego "Barrer el suelo")	*Usuario pulsa y mantiene hacia una dirección del joystick táctil; ejecución del método Update() del controlador de juego; asset escoba.*	*La escoba se mueve con una velocidad constante hacia la dirección en la que el usuario pulsa el joystick*	La esperada.

Tabla 4.1: Tabla de pruebas unitarias de caja negra ejemplares de CleanQuest

4.1.2 Pruebas unitarias de caja blanca

Las pruebas unitarias de tipo caja blanca realizadas, a diferencia de las de caja negra, son aquellas en las que, al ejecutar una "Acción", se comprueban todos los posibles caminos que puede tomar la ejecución normal de la misma.

Como este tipo de pruebas son bastante más complejas que las de caja negra y siguiendo con la decisión de sólo reflejar las pruebas más representativas realizadas, en ésta memoria sólo se reflejará una prueba unitaria de caja blanca, a modo de ejemplo de las realizadas.

Para realizar este tipo de pruebas se utilizó el "criterio de cobertura de caminos" [49], adaptándolo a la condición de una "Acción". Mediante este criterio se tienen en cuenta casos de prueba suficientes para que se ejecuten todos los caminos, en este caso, de un método dentro de una "Acción". Entendiendo camino como una secuencia de sentencias encadenadas desde la entrada del método de la "Acción" hasta su salida.

La prueba unitaria de caja blanca que se utilizará como ejemplo, será la realizada a la "Acción" "Prueba mecanismo *drag and drop"* de la tabla 4.1, sobre su método *Update()*. A continuación se muestran las 3 fases del proceso de extracción del grafo del método elegido, necesario para la cobertura de caminos: El código del método en sí, el diagrama de flujo surgido del código y el grafo de caminos surgido del diagrama de flujo.

```
function Update(){

    if (Input.GetMouseButton(0)){  // si han dejado pulsada la pantalla
        // lanza un rayo en la pulsacion
        var ray = Camera.main.ScreenPointToRay(Input.mousePosition);
        if (!dragObj ){  // si nada estaba cogido...
            // y el rayo alcanza a un rigidbody
            if (Physics.Raycast(ray, hit) && hit.rigidbody){
                dragObj = hit.transform;  // guarda el transform del rigidbody
                length = hit.distance;  // y la distancia desde la pulsacion
            }
        }
        else {  // si algo ya estaba cogido...
            // calcular la velocidad necesaria para seguir la pulsacion
            var vel = (ray.GetPoint(length) - dragObj.position) * speed;
            // limitar la velocidad maxima para evitar atravesar objetos
            if (vel.magnitude > maxSpeed) vel *= maxSpeed / vel.magnitude;
            // establecer la velocidad
            dragObj.rigidbody.velocity = vel;
        }
    }
    else {  // si no han dejado pulsada la pantalla
        dragObj = null;  // libera el objeto cogido (si lo tenia)
    }
}
```

Figura 4.1: Método Update() de la "Acción" Prueba del mecanismo drag and drop

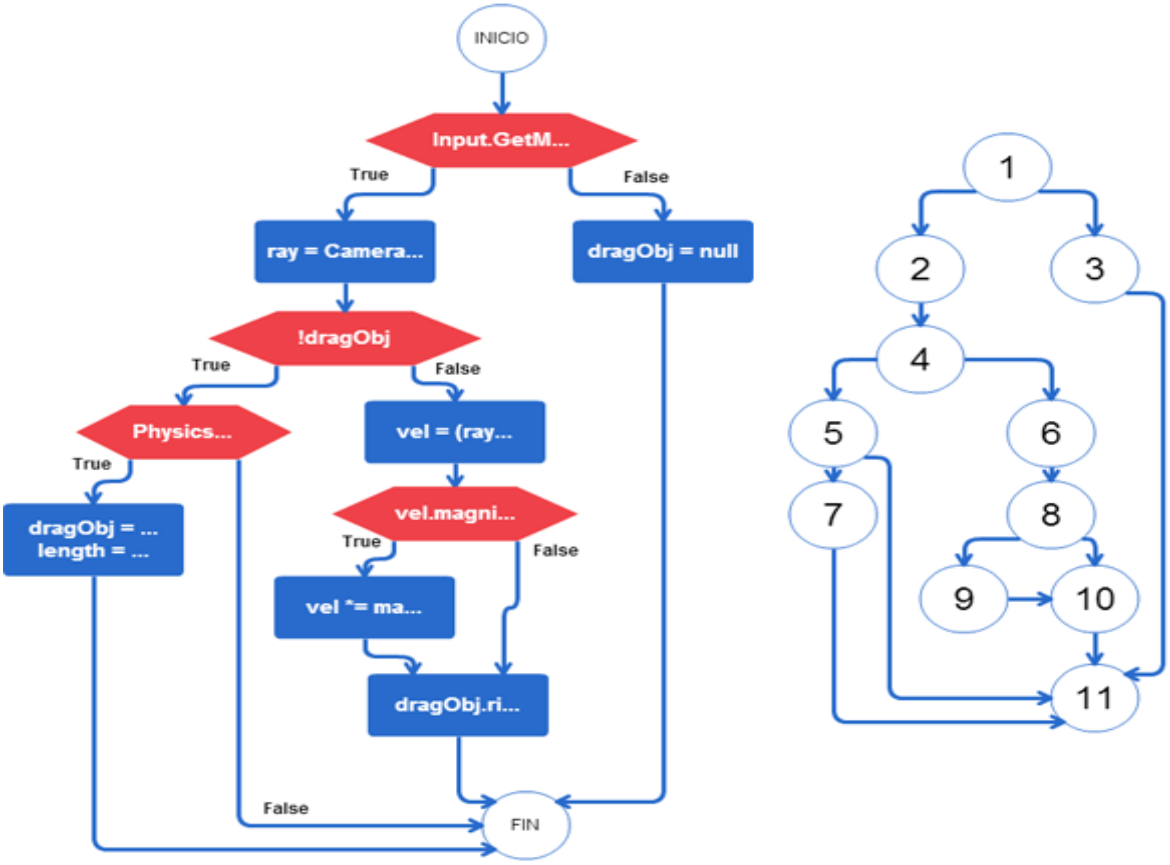


Figura 4.2: Diagrama de flujo y grafo de caminos del método Update() de la "Acción"

Una vez extraídos el diagrama de flujo y el grafo de caminos de la acción, se procede a hallar el número de caminos independientes de la "Acción" a partir del grafo de caminos. El objetivo es hallar el número de casos de prueba que se realizarán como pruebas de caja blanca a partir de estos caminos. Para ello, inicialmente es necesario calcular la complejidad ciclomática del grafo de caminos. Existen varios métodos válidos para calcular la complejidad ciclomática; en este caso, se decide exponer todos para corroborar que coinciden y que por lo tanto el grafo de caminos está bien realizado:

- Número de regiones del grafo = 5
- Aristas - nodos + 2 = 14 - 11 + 2 = 5
- Nodos predicado + 1 = 4 + 1 = 5

Los resultados indican que se puede establecer la complejidad ciclomática del grafo es 5.

Como el código inicial y el grafo resultante no son complejos y es sencillo identificar los posibles caminos independientes a simple vista, atendiendo a las heurísticas del criterio de cobertura de caminos se puede establecer el número de caminos independientes como equivalente a la complejidad ciclomática, 5. Igualmente este número es equivalente a los casos de prueba que se necesitarán para esta prueba de caja blanca.

Teniendo esto en cuenta se han diseñado las siguientes pruebas en base a cada camino independiente que forman en método, presentadas en forma de tabla:

Camino independiente	Salida esperada	Salida real

1-2-4-5-7-11	ray = Camera.main…;
	dragObj = hit.transform;
	length = hit.distance;
		La esperada

1-2-4-5-11	ray = Camera.main…;	La esperada

1-2-4-6-8-9-10-11	ray = Camera.main…;
	vel = (ray.GetPoint(length)…;
	vel *= maxSpeed/vel.magnitude;
	dragObj.rigidbody.velocity = vel;
		La esperada

1-2-4-6-8-10-11	ray = Camera.main…; vel = (ray.GetPoint(length)…; dragObj.rigidbody.velocity = vel;	La esperada
1-3-11	dragObj = null;	La esperada

Tabla 4.2: Tabla de pruebas de caja blanca de la "Acción" Prueba del mecanismo drag and drop

4.2 Pruebas de integración

Las pruebas de integración son aquellas cuyo objetivo es evaluar una serie de pruebas unitarias interrelacionadas, esto es, evalúan un conjunto de métodos que trabajan de forma conjunta para formar un módulo de funcionalidad (proceso) de mayor tamaño dentro del sistema.

Continuando con la filosofía del apartado de las pruebas unitarias (apartado 4.1), para este tipo de pruebas se siguen considerando "Acciones" en lugar de solamente métodos. De esta forma, se decide que como pruebas de integración se evalúen un conjunto de pruebas unitarias sobre "Acciones" interrelacionadas lo suficientemente sustancial. Este conjunto pasa a considerarse como **cada pantalla completa que conforma el videojuego**. En otras palabras, en cada prueba de integración se evalúa la correcta funcionalidad de todas las pruebas unitarias que conforman una pantalla individual de *CleanQuest*, trabajando en conjunto. Por lo tanto, el objetivo de cada una de estas pruebas es por un lado, evaluar todas las pruebas unitarias que forman una pantalla individual y por otro, de forma derivada, que el funcionamiento global de esa pantalla coincida con el descrito en la pantalla análoga del prototipo de bajo nivel de fidelidad propuesto (apartado 3.4.1).

Debido a que este tipo de pruebas no son más que un conjunto de pruebas unitarias, ya mostradas de forma ejemplar en el apartado anterior, en este apartado sólo se mostrará una de ellas a modo ejemplo, concretamente la que evalúa la pantalla correspondiente al menú principal (MP). Este tipo de pruebas se documentaron mediante una tabla individual para cada una con el siguiente formato:

- **Pantalla.** Identificador acompañado del nombre común de la pantalla a evaluar atendiendo al prototipo de bajo nivel de fidelidad.
- **Unitarias Blancas y Negras.** *Checkbox* que confirma si el conjunto de las pruebas unitarias de caja blanca y negra (respectivamente) han sido validadas.
- **Resultado esperado.** Funcionalidad de la página de acuerdo a su propósito en el prototipo de bajo nivel de fidelidad
- **Salida real.** Listado de los eventos (incluyendo errores) o retroalimentación que realmente han ocurrido en la prueba.

Pantalla: MP (Menú principal)			
Unitarias Blancas	**OK**	**Unitarias Negras**	**OK**
Resultado esperado: Menú principal del juego. Permite el acceso al resto de secciones del juego y es el punto de retorno al salir de cada una de ellas. Diseñado para ser lo más intuitivo posible. Presenta un fondo transparente con un modelo de la habitación de Jaimito 3d de fondo, rotando sobre sí misma lentamente.			
Resultado real: – El esperado.			

Tabla 4.3: Ejemplo de prueba de integración

4.3 Pruebas de sistema

Las pruebas sistema son aquellas cuyo objetivo es comprobar y validar la funcionalidad del sistema completo una vez implementado o de módulos del mismo lo suficientemente grandes e independientes. De igual forma podrían considerarse como pruebas que evalúan una serie de pruebas de integración interrelacionadas. La realización de estas pruebas se divide en dos sub-fases: pruebas de subsistemas y de sistema completo.

Las **pruebas de subsistemas y sistema completo**, son Pruebas que se realizan una vez el prototipo software de alto nivel de fidelidad esté completamente implementado. Se evalúa primeramente la correcta funcionalidad de cada subsistema aislado que conforma el sistema completo. En este caso, cada subsistema se considera como un conjuntos de prueba de integración (pruebas con pantallas individuales) que representan un aspecto determinado del juego. A continuación se evalúa la funcionalidad global del sistema completo, esto es, con todos los subsistemas integrados y trabajando conjuntamente.

4.3.1 Pruebas de subsistemas y de sistema completo

Las pruebas de subsistemas y sistema completo se realizan una vez que el prototipo software de alto nivel de fidelidad está completamente implementado. Se evalúan de forma asilada sus subsistemas y una vez validados se integran, formando el sistema completo (el videojuego) y se evalúa como tal.

Pruebas de subsistemas

En *CleanQuest*, una prueba de subsistema se encarga de evaluar una serie de pruebas de integración interrelacionadas que trabajan en conjunto para representar la funcionalidad de un aspecto concreto del juego o subsistema aislado.

Como se estableció en el apartado de “Pruebas de integración” (apartado 4.2), las pruebas de integración evalúan pantallas individuales del juego. Por lo tanto, una prueba de subsistema evaluará la funcionalidad completa de un grupo de pantallas que trabajen entre sí, formando un aspecto del juego o subsistema. Teniendo esto en cuenta, para realizar las pruebas de subsistemas, se consideraron los siguientes aspectos del juego o subsistemas a evaluar, junto con el grupo de pantallas que lo conforma, atendiendo a los grupos de pantallas diferenciados en el prototipo de bajo nivel de fidelidad (apartado 3.4.1):

- Subsistema “Control parental”: Grupo de pantallas de control parental.
- Subsistema “Galería de cromos”: Grupo de pantallas de la galería de cromos.
- Subsistema “Mini-juegos”: Grupos de pantallas del menú principal + Grupo de pantallas de mini-juegos genéricas + Grupo de pantallas de jugabilidad en mini-juegos + Grupo de pantallas de presentación del desafío de realidad aumentada.

Las pruebas de subsistema realizadas sobre cada uno de estos 3 subsistemas evalúan funcionalidades y características que tengan que ver **únicamente con la interrelación entre las pantallas que los conforman**, ya que las pantallas como tales, han sido validadas en las pruebas de integración anteriores. Estas pruebas son específicas de cada subsistema y son notablemente diferentes entre subsistemas, lo que dificultó la extrapolación de las pruebas validadas de un subsistema a otros. Debido a esto el número de pruebas de subsistema realizadas aumentó en gran medida, con escasas pruebas ejemplares que extrapolar. Por lo tanto, en esta memoria se decide mostrar sólo una prueba de subsistema de las realizadas, a modo de ejemplo, y así mostrar igualmente el procedimiento general y el formato de tabla individual que se utilizó en este tipo de pruebas:

- **ID.** Número de 3 dígitos que identifica unívocamente la prueba de sistema.
- **Nombre.** Nombre de la pantalla a modo de descripción breve de la función que evalúa.
- **Subsistema.** Subsistema sobre el que se realiza la prueba. Si es necesario puede incluir la pantalla concreta de ese subsistema entre paréntesis.
- **Pasos para la ejecución.** Listado de pasos que deben realizarse para ejecutar la funcionalidad del subsistema que debe ser evaluada.
- **Resultado esperado.** Listado de los eventos o retroalimentación esperados que deben ocurrir en la prueba.
- **Salida real.** Listado de los eventos (incluyendo errores) o retroalimentación que realmente han ocurrido en la prueba.

ID: 001	**Nombre:** Paso entre pantallas del CP	**Subsistema:** “Control Parental” (CP-1)
Pasos para la ejecución: – Usuario inicia el juego por primera vez. – Usuario pulsa el botón con el símbolo siguiente		
Resultado esperado: – Paso a la pantalla CP-2 acompañada del sonido correspondiente.		
Resultado real: – El esperado.		

Tabla 4.4: Ejemplo de prueba de subsistema

Pruebas de sistema completo

Una vez validadas las pruebas de subsistemas de forma aislada, éstos se integran adecuadamente hasta formar el sistema completo, esto es, el prototipo de software de alto nivel de fidelidad, y se realizan pruebas de sistema completo sobre el mismo.

Estas pruebas, debido a que todas las pruebas de cada subsistema ya han sido validadas, sólo se centran en evaluar las funcionalidades concretas que atañen a las

interrelaciones entre subsistemas. En este caso, las funcionalidades que representan interrelaciones entre subsistemas son pocas y pueden resumirse en dos puntos:

- Navegación entre pantallas de un subsistema a otro.
- Persistencia en general.

El primero resulta evidente, ya que es esencial para la integración. El segundo se refiere a la comunicación entre subsistemas mediante datos persistentes, independientemente de la pantalla o subsistema, o del reinicio de la aplicación. Este segundo punto puede evaluarse considerando las siguientes pruebas ejemplares:

- **Cromos coleccionables.** Cada vez que el usuario consiga un cromo coleccionable en cualquier pantalla debe aparecer en la galería de cromos.

- **Introducción de códigos/contraseñas.** Cuando el usuario introduzca un código en el control parental para el "Desafío de realidad aumentada", si en la galería de códigos introduce el mismo código, debe desbloquearse el cromo correspondiente.

- **Inicio correcto.** Una vez que el usuario complete el control parental, cada vez que se vuelva a iniciar la aplicación debe iniciarse en el menú principal.

Debido a que estas pruebas ejemplares son evaluadas mediante un procedimiento y formato de tabla muy similar a las de las pruebas de subsistemas (tabla 4.3), se decide únicamente reflejar una prueba de sistema completo a modo de ejemplo:

ID: S-001	**Nombre:** Inicio correcto del juego	**Subsistemas:** "Control Parental" y "Menú Principal"
Pasos para la ejecución: – Usuario inicia el juego por primera vez. – Usuario completa el control parental. – Usuario inicial el juego por segunda vez.		
Resultado esperado: – El juego se inicia por segunda vez en la pantalla MP del subsistema Menú Principal		
Resultado real: – El esperado.		

Tabla 4.5: Ejemplo de prueba de sistema completo

Una vez validadas en su totalidad las pruebas de sistema completo, es posible confirmar la correcta funcionalidad global del prototipo de software de alto nivel de fidelidad. Con este prototipo completado es posible proporcionárselo a los usuarios con los que se realizarán las pruebas de validación de producto.

4.4 Validación del producto

Las pruebas de validación del producto se realizan una vez que se haya validado el prototipo software de alto nivel de fidelidad, ya que se realizan con el mismo, a modo de pruebas definitivas, para verificarlo. Se caracterizan por que utilizan usuarios reales objetivo y evalúan su experiencia de uso con el prototipo. Debido a su naturaleza, **este tipo de pruebas son las más importantes de todo el proceso de desarrollo**, ya que, una vez confirmada la correcta funcionalidad del prototipo en las pruebas anteriores aisladas, estas pruebas son las únicas capaces de comprobar su efectividad a la hora de de cumplir su propósito establecido: solucionar el problema de desobediencia o indiferencia infantil en la realización de las tareas domésticas de limpieza del hogar.

En este caso se contó con la participación de **13 niños (8 niñas y 5 niños)** de entre 5-10 años de edad (cercano al rango de edad objetivo, de 6-9 años), junto con uno de los padres o tutores de cada niño, haciendo **9 padres o tutores en total**, ya que algunos de los niños eran hermanos. El periodo en el que se realizaron las pruebas abarca desde el 25 de diciembre del 2014 hasta el 6 de enero del 2015, periodo de vacaciones para los niños.

Estas pruebas se englobaron en **2 etapas**, ambas en un entorno no controlado, el entorno doméstico de cada niño, y con un procedimiento de monitorización (por parte de los padres o tutores) y recopilación de opiniones al final de cada etapa mediante un breve cuestionario para evaluarlas. La primera etapa de las pruebas tenía como objetivo **valorar la experiencia general** de los usuarios y los padres o tutores de los mismos con el juego, mientras que la segunda etapa tenía como objetivo **valorar la efectividad del juego** para solucionar el problema establecido. Teniendo esto en cuenta, a continuación se comentan punto por punto (a modo de registro) los acontecimientos más importantes que sucedieron durante las dos etapas del periodo de pruebas, para así mostrar una visión general de cómo se realizó todo este proceso de pruebas y en qué consistió:

1ª Etapa: Valoración de la experiencia con el juego en general

- **25/12/2014.** Se reunió a los participantes (tanto a los niños como sus padres o tutores) y se les proporcionó una copia de *CleanQuest* en sus dispositivos *Android* si los poseían, en caso contrario se les prestaba un dispositivo *Android* con el juego ya instalado. A continuación se les comunicó verbalmente a los padres o tutores una serie de indicaciones generales sobre esta etapa, apuntado que, si les fuera posible, monitorizaran sutilmente a sus hijos mientras utilizaban el juego. Igualmente se les proporcionó un manual de usuario, por si encontraban algún tipo de dificultad en su uso del juego.

- **31/12/2014.** Se reunió a los participantes nuevamente y se les proporcionó un cuestionario sobre la primera etapa, con una parte específica para padres o tutores y otra para niños, con el objetivo de valorar la experiencia general sobre el juego.

2ª Etapa: Valoración de la efectividad del juego

- **31/12/2014.** En el mismo día en que finalizó la primera etapa, una vez que los participantes terminaron el cuestionario de esta etapa, se les preguntó a los padres o tutores si alguno de sus hijos había completado ya en la primera etapa algún desafío de realidad aumentada para obtener nuevos cromos. A los que respondieron afirmativamente se les proporcionó directamente el cuestionario de la segunda etapa, nuevamente con una parte específica para padres o tutores y otra para niños. Los que realizaron el cuestionario de la segunda etapa en ese momento completaron el periodo de pruebas y abandonaron el proceso. Al resto de participantes, concretamente a los niños, simplemente se les instó para que en esta segunda etapa completaran las 3 fases de un mini-juego. El objetivo, no revelado a los niños (sólo a los padres o tutores), era que el juego les informara de algún desafío de realidad aumentada y comprobar cómo reaccionaban los niños al mismo.

- **06/01/2015.** Se reunió nuevamente a los participantes restantes y se les proporcionó un cuestionario sobre la segunda etapa (el mismo que el que realizaron los participantes que completaron prematuramente el proceso de pruebas), con una parte específica para padres o tutores y otra para niños, con el objetivo de valorar la efectividad del juego para resolver el problema planteado. Con ello, todos los participantes terminaron la segunda etapa y se dio por finalizado el periodo de pruebas en general.

Resultados de las pruebas de validación de producto

Los resultados de este tipo de pruebas pueden interpretarse a partir de los resultados de los cuestionarios de cada etapa. A continuación se muestran los resultados de ambos cuestionarios en forma de tablas, incluyendo las preguntas realizadas en cada cuestionario, sus posibles respuestas y la opción más popular entre ellas:

***CleanQuest* - Cuestionario final de 1ª Etapa**		
Parte específica para padres o tutores		
Cuestión	**Rango de opciones**	**Opción más popular (aprox.)**
He podido navegar por el control parental sin dificultad.	1. Muy en desacuerdo; 2. En desacuerdo; 3. Indiferente; 4. De acuerdo; 5. Completamente de acuerdo.	5. Completamente de acuerdo; con un 100% de popularidad.
He entendido el video introductorio del control parental sin dificultad.	Ídem	5. Completamente de acuerdo; con un 70% de popularidad.
He podido introducir los datos necesarios para el desafío de realidad aumentada sin dificultad en el control parental.	Ídem	4. De acuerdo; con un 60% de popularidad.
No he necesitado el manual de usuario para jugar al juego.	Ídem	5. Completamente de acuerdo; con un 90% de popularidad.

Parte específica para niños		
Cuestión	**Rango de opciones**	**Opción más popular**
He podido navegar por las pantallas sin dificultades.	1. Muy en desacuerdo; 2. En desacuerdo; 3. Indiferente; 4. De acuerdo; 5. Completamente de acuerdo.	5. Completamente de acuerdo; con un 80% de popularidad.
He entendido la función de cada botón en cada pantalla sólo con verlos.	Ídem	5. Completamente de acuerdo; con un 100% de popularidad.
He podido ver los cromos obtenidos en la galería de cromos sin dificultad.	Ídem	4. De acuerdo; con un 50% de popularidad.
Me gustan los cromos de *CleanQuest* y quiero conseguir más	Ídem	5. Completamente de acuerdo; con un 100% de popularidad.
He podido acceder a los mini-juegos sin dificultades.	Ídem	5. Completamente de acuerdo; con un 100% de popularidad.
El mini-juego “Recoger los juguetes” me ha parecido difícil.	Ídem	1. Muy en desacuerdo; con un 100% de popularidad.
El mini-juego “Recoger los juguetes” me ha parecido divertido.	Ídem	5. Completamente de acuerdo; con un 100% de popularidad.
El mini-juego “Barrer el suelo” me ha parecido difícil.	Ídem	2. En desacuerdo; con un 50% de popularidad.
El mini-juego “Barrer el suelo” me ha parecido divertido.	Ídem	4. De acuerdo; con un 100% de popularidad.
El mini-juego “Limpiar los cristales” me ha parecido difícil.	Ídem	1. Muy en desacuerdo; con un 100% de popularidad.
El mini-juego “Limpiar los cristales” me ha parecido divertido.	Ídem	4. De acuerdo; con un 80% de popularidad.
Me ha gustado ayudar a la familia de Jaimito a realizar las tareas de limpieza.	Ídem	5. Completamente de acuerdo; con un 100% de popularidad.

Tabla 4.6: Resultados del cuestionario de la 1ª etapa de las pruebas de validación de producto de CleanQuest.

***CleanQuest* - Cuestionario final de 2ª Etapa**		
Parte específica para padres o tutores		
Cuestión	**Rango de opciones**	**Opción más popular**
Mi hijo ha realizado la tarea de limpieza en la vida real que se le ha encomendado en el desafío de realidad aumentada.	1. Si; 2. No.	1. Sí; con un 100% de popularidad.
Mi hijo ha realizado la tarea de limpieza encomendada en el lugar indicado, por propia voluntad y contento.	1. Muy en desacuerdo; 2. En desacuerdo; 3. Indiferente; 4. De acuerdo; 5. Completamente de acuerdo.	5. Completamente de acuerdo; con un 90% de popularidad.
He podido introducir el código necesario para desbloquear el	Ídem	5. Completamente de acuerdo; con un 80% de popularidad.

cromo de premio por completar el desafío de realidad aumentada en la galería de cromos sin dificultad.		
Parte específica para niños		
Cuestión	**Rango de opciones**	**Opción más popular**
He realizado la tarea de limpieza en la vida real que se me ha encomendado en el desafío de realidad aumentada.	1. Si; 2. No.	1. Sí; con un 100% de popularidad.
He realizado la tarea de limpieza encomendada en el lugar indicado, por propia voluntad y contento.	1. Muy en desacuerdo; 2. En desacuerdo; 3. Indiferente; 4. De acuerdo; 5. Completamente de acuerdo.	5. Completamente de acuerdo; con un 100% de popularidad.
Me han gustado los cromos obtenidos al completar el desafío de realidad aumentada.	Ídem	5. Completamente de acuerdo; con un 80% de popularidad.
Estoy dispuesto/a a realizar más tareas de limpieza a cambio de cromos coleccionables.	Ídem	5. Completamente de acuerdo; con un 100% de popularidad.
Estoy dispuesto/a a ayudar más en las tareas de limpieza de casa en general.	Ídem	5. Completamente de acuerdo; con un 80% de popularidad.

Tabla 4.7: Resultados del cuestionario de la 2ª etapa de las pruebas de validación de producto de CleanQuest.

Como puede observarse estos cuestionarios han sido diseñados pensando en el rango de edad del usuario objetivo (niños de 6-9 años): breves y con preguntas concisas para mantener la atención de los usuarios más pequeños. Igualmente intentando de forma paralela cubrir todas las cuestiones que reflejen en su totalidad la experiencia del usuario con el juego. Cabe destacar que, incluso con estos simples cuestionarios, se necesitó reformular ciertas cuestiones para que algunos de los niños las entendiesen sin problemas.

Respecto a los resultados de ambos cuestionarios, es evidente que reflejan una más que positiva aceptación del juego por parte de los participantes. **Aunque estadísticamente no son significativos**, debido al grupo reducido de participantes, **de forma explorativa** (sobre todo atendiendo al segundo cuestionario) **apuntan a que el juego efectivamente cumple su propósito como herramienta para solucionar el problema planteado tanto a corto como a largo plazo.**

De forma adicional, se incluyen una serie de videos que enviaron algunos padres, resultado de la monitorización durante la primera etapa de estas pruebas. Antes de incluir estos vídeos en esta memoria, los padres involucrados rellenaron y firmaron un documento por el cual autorizaban la publicación de los mismos, regulado por el artículo 18 de la constitución española [50], la "*Ley 1/1982 de 5 de mayo referida al derecho al honor, a la intimidad personal y a la propia imagen*" [51] y la "*Ley 5/1999 de 13 de diciembre sobre la Protección de Datos de Carácter Personal (LOPD)"* [52].

<https://www.youtube.com/watch?v=Ya51OXaoqLI>.

<https://www.youtube.com/watch?v=GnbobquWqIc>.

<https://www.youtube.com/watch?v=ZQg4MdxNgG4>.

<https://www.youtube.com/watch?v=LfC5z857UMY>.

<https://www.youtube.com/watch?v=fZ6_jVc5svQ>.

<https://www.youtube.com/watch?v=M01YJdUhvk8>.

<https://www.youtube.com/watch?v=P1KOTu2di4Y>.

<https://www.youtube.com/watch?v=HDIKUZCwkL8>.

Con estos videos se obtuvieron observaciones cualitativas extra del uso del juego, aparte de los resultados de carácter cuantitativo obtenidos a partir de los cuestionarios. Como estos vídeos en un principio no formaban parte de estas pruebas de validación (fue una iniciativa de los padres participantes), no se tendrán en cuenta a la hora de interpretar los resultados en esta fase, pero sí se realizará un inciso refiriéndose a ellos en las líneas futuras de *CleanQuest* (apartado 5.2).

5. Conclusiones y líneas futuras

En este apartado se exponen las conclusiones y líneas futuras de todo el proceso de este TFG. Se habla de las **conclusiones** en general y enfocándolas **a los objetivos establecidos en la introducción**, esto es, justificando como se han alcanzado estos objetivos a través del proceso. Igualmente se proponen **líneas futuras** para el TFG, como posibles mejoras del prototipo de SW, nociones para su mantenimiento posterior, etc.

5.1 Conclusiones

El fin de este TFG era desarrollar una herramienta, basada en técnicas de videojuegos, cuyo uso pudiera solucionar el problema de la desobediencia o indiferencia infantil en la colaboración en las tareas de limpieza del hogar.

Para confirmar que verdaderamente se ha alcanzado el fin original a través del proceso de desarrollo, es necesario comprobar que todos los objetivos establecidos en el apartado de "Introducción" (apartado 1), máximas de todo el proceso, han sido satisfechos. A continuación se listan cada uno de estos objetivos, junto con la explicación de cómo y en qué puntos del proceso se han cumplido. Como objetivos principales se determinaron:

- ***Desarrollar un videojuego que pueda ser utilizado como herramienta para resolver el problema de la desobediencia o indiferencia infantil ante la colaboración en las tareas de limpieza del hogar de una forma óptima (a corto y a largo plazo).***

Objetivo que engloba todo el proceso de este TFG. Puede ser considerado como alcanzado atendiendo a dos puntos clave dentro del proceso de desarrollo.

El primero se corresponde con los apartados "Estado del arte" (apartado 2) y "Desarrollo" (apartado 3). En el "Estado del arte" se establecieron el género y las características base de *CleanQuest* a través del análisis de las características de otros productos de software actuales que solucionan o ayudan a solucionar el mismo problema. ***CleanQuest* recoge gran parte de sus virtudes para solucionar el problema tanto a corto plazo**, mediante el uso de cromos virtuales coleccionables como recompensa por realizar tareas de limpieza del hogar en la vida real (todo lo referente al desafío de realidad aumentada); **como a largo plazo**, mediante la ambientación y el trasfondo del juego, que logran presentar estas tareas de forma positiva a los usuarios del rango de edad objetivo, esto es, como algo necesario que hay que hacer para hacer feliz al resto de los familiares del hogar, fomentando la responsabilidad.

En el apartado "Desarrollo", partiendo de las características base de *CleanQuest*, primeramente se refinaron en la fase de preproducción, aumentando el atractivo del juego hacia el usuario objetivo. A continuación se **desarrolló** el videojuego en su totalidad, resultando en el prototipo de software de alto nivel de fidelidad.

Por otro lado, es necesario confirmar todas estas virtudes atribuidas en el primer punto. Esto se realiza en el segundo punto del proceso, correspondiente a la fase de pruebas (apartado 4), donde se **valida la funcionalidad del juego y su propósito** utilizando el prototipo de software de alto nivel de fidelidad, fruto del desarrollo. Los resultados de los cuestionarios realizados al grupo de usuarios utilizado en las pruebas de validación del producto reflejaron, **de forma explorativa**, que efectivamente el juego cumple su propósito como herramienta. Aunque, como ya se comentó en el apartado de "Validación de producto" (apartado 4.4), debido al reducido número de participantes utilizados, éstas pruebas no son estadísticamente

significativas y para confirmar definitivamente sus resultados habría que utilizar un tamaño de muestra mucho más grande.

- ***Seleccionar los medios y metodologías óptimos y el sistema adecuado que soporte el videojuego, para el desarrollo.***

Este objetivo es satisfecho atendiendo a los apartados de "Metodología y medios empleados" (apartado 1.4), "Análisis del estado del arte del sistema y del kit de desarrollo de videojuegos" (apartado 2.3) y "Selección de herramientas secundarias" (apartado 3.4.2).

En el apartado de "Metodología y medios empleados" se estableció la **metodología** utilizada en el desarrollo de *CleanQuest*, combinando varias metodologías existentes en el desarrollo de productos de software adaptadas para cada uno de los aspectos principales del desarrollo del videojuego. Posteriormente en el apartado de "Análisis del estado del arte del sistema y del kit de desarrollo de videojuegos" se compararon los principales **sistemas** y kits de desarrollo de videojuegos existentes en el mercado y se seleccionan los más adecuados atendiendo a la naturaleza de *CleanQuest*, justificando su uso. Por último, en el sub-apartado de la preproducción "Selección de herramientas secundarias" se seleccionaron los programas auxiliares al kit de desarrollo de videojuegos más adecuados atendiendo a las necesidades del desarrollo surgidas en la preproducción, completando de esta manera la lista de **medios** que se utilizaron en el desarrollo.

- ***Crear un plan de distribución adecuado para el videojuego, permitiendo que por un lado pueda cumplir su propósito principal (resolver el problema) sin inconvenientes y por otro su monetización, explotando el creciente mercado de los videojuegos educativos.***

Este objetivo se cumplió en los apartados "Modelo de negocio base" (apartado 2.2.7) y "Postproducción" (apartado 3.8). En el primero se planteó un modelo de negocio para *CleanQuest*, basado en la distribución gratuita de una versión básica del videojuego a través de Google Play, con la posibilidad de adquirir pequeñas expansiones opcionales de contenido mediante micro pagos. Como ya se justificó en el apartado, al ser gratuito puede **cumplir su objetivo sin problemas**, haciéndolo incluso más atractivo al usuario objetivo, mientras que con los micro pagos **se asegura el beneficio monetario**. En el segundo apartado mencionado, ya con el prototipo software de alto nivel de fidelidad presente, se confirmó este modelo de negocio como definitivo. Por supuesto, como ya se mencionó en la "Postproducción", este modelo de negocio no deja de ser teórico y por lo tanto, para demostrar su viabilidad real, se necesitaría ponerlo en práctica mediante la comercialización real del videojuego.

Como objetivos secundarios se determinaron:

- ***Adquirir experiencia como desarrollador de videojuegos de cara al mundo laboral.***

Como ya se comentó en el apartado “Fases del proceso” (apartado 1.5), en el proceso de desarrollo de este TFG se ha incluido un proceso de desarrollo de videojuegos similar al utilizado en la industria profesional (obviamente simplificado debido a la pequeña escala del proyecto en comparación). El hecho de haber realizado todas estas fases cumpliendo el resto de objetivos, es suficiente para demostrar que este objetivo ha sido satisfecho.

Como se puede observar, todos los objetivos establecidos han sido cumplidos, por lo que el desarrollo de este TFG puede ser considerado como éxito. Sin embargo, todavía pueden aplicarse ciertas mejoras a *CleanQuest*, que permitirán asegurar su continuidad. Estas serán expuestas en el apartado “Líneas futuras”.

5.1.1 Valoración personal

“Una vez finalizado todo el desarrollo puedo aportar mi valoración personal. Durante todo este proceso he tenido que utilizar la mayor parte de mi experiencia como estudiante de informática, haciendo uso tanto de competencias adquiridas en diversas asignaturas de la carrera, como de nuevas competencias adquiridas durante el propio desarrollo. El haber llevado a cabo un TFG como este me ha hecho ver cómo es en realidad el mundo profesional de los videojuegos y lo que acarrea, pero lejos de haber sido una experiencia negativa me ha motivado para probar suerte en el mundillo.

Respecto al resultado final de todo el desarrollo, CleanQuest, no puedo estar más orgulloso. Especialmente sabiendo que me he mantenido fiel a mis principios en lo que respecta a la creación de assets; he conseguido crear desde 0 la mayor parte de los assets utilizados para el juego, desde los modelos 3d hasta la interfaz (a excepción de las pistas de audio). Esto significa que he conseguido dar uso mis habilidades artísticas utilizadas hasta ahora como hobby, en conjunción con todas las competencias anteriormente mencionadas. Igualmente me siento orgulloso de haber creado un juego que pueda ser utilizado como herramienta para solucionar un problema social tan importante, aportando así mi granito de arena para crear un mejor ambiente doméstico en la sociedad actual.”

5.2 Líneas futuras

En este apartado se exponen las líneas futuras para *CleanQuest*, videojuego fruto del proceso de desarrollo de todo este TFG. Estas líneas tratan de los aspectos necesarios que permitan la continuidad de *CleanQuest*, una vez se dé por finalizado el TFG, como posibles mejoras del prototipo de SW, nociones para su mantenimiento posterior, etc. A continuación se lista cada una de ellas acompañadas de una pequeña justificación:

- **Aplicar el modelo de negocio y el plan de gestión de parches y actualizaciones establecidos en la fase de postproducción.** Como ya se mencionó en el apartado de "Postproducción", se decidió que el modelo de negocio y la gestión de parches y actualizaciones planteados fuesen sólo teóricos, esto es, no verdaderamente realizados en este TFG, sólo planteados. El próximo paso lógico después de validar el prototipo de software de alto nivel de fidelidad, ya considerado como producto final, sería comercializarlo en base al modelo de negocio establecido en este apartado y mantenerlo al día con el plan de gestión de parches y actualizaciones del mismo. En otras palabras, una vez terminado el videojuego, sería necesario poner en práctica el modelo de negocio y el plan de gestión de parches y actualizaciones planteados para el mismo.

- **Optimización de modelos 3d.** Todos los modelos 3d de *CleanQuest*, presentan un estilo (dictado por la dirección artística) *low poly*. Esto quiere decir que los modelos son ligeros, compuestos por un bajo número de polígonos y texturas simples. Se decidió usar este estilo debido a la condición de *CleanQuest* como videojuego para dispositivos móviles con sistema operativo *Android*, ya que la potencia gráfica de estos dispositivos suele ser baja y es necesario utilizar este estilo de modelos 3d para garantizar la fluidez del juego. Aun así, el número de polígonos utilizados en los modelos de *CleanQuest* puede ser reducido aun más, todavía manteniendo su aspecto original y optimizando de esta manera los modelos y el juego en general. Esta tarea no es fácil y requiere de cierto nivel de maestría en el modelado 3d de la que no se disponía en el momento del desarrollo, dejando este punto como tarea para el futuro que convenientemente podría distribuirse como un parche o actualización del videojuego.

- **Ocultar la introducción del código/contraseña en del control parental.** De cara a la creación del prototipo de software de alto nivel de fidelidad (para facilitar las pruebas) así como por petición de algunos de los participantes de las pruebas de validación de producto (apartado 4.4) se decidió no ocultar (sustituir por asteriscos) la introducción de los códigos de confirmación para las tareas del desafío de realidad aumentada dentro del control parental. Por supuesto, antes de comercializar el videojuego, sería necesario ocultarlas tanto en el juego como en el video introductorio del control parental o implementar alguna funcionalidad que permitirá decidir al usuario si desea ocultar su introducción o no (algún tipo de *checkbox* indicativo).

- **Añadir un sistema de recuperación de códigos/contraseñas avanzado.** En el prototipo de bajo nivel de fidelidad de *CleanQuest* (apartado 3.4.1), en la pantalla *(Sub) GC-IC* o Sub-pantalla de introducción de códigos/contraseñas de la galería de cromos, se incluyó un botón para resetear el control parental de juego. La finalidad de este botón era volver a las pantallas del control parental e introducir nuevamente todos los datos necesarios del control parental (códigos/contraseñas y lugares de las tareas referentes al desafío de realidad aumentada), en caso de olvido o necesidad de cambio de los originariamente introducidos. El caso es que para ello se necesitaría algún tipo de control de acceso extra para garantizar que sólo los padres o tutores del jugador puedan acceder a esta funcionalidad. Para implementar esta funcionalidad, sin recurrir a la eliminación del progreso del usuario (reinstalación de la aplicación), sería necesario implementar un sistema similar al sistema común de recuperación de contraseña olvidada en las webs que requieren el registro de un usuario, utilizando la dirección de correo electrónico de los padres o tutores para corroborar su identidad. Esto hubiera aumentado de forma drástica la dificultad del desarrollo, ya que requeriría ligar el videojuego a un servidor en línea que guardara los correos electrónicos de todos estos padres o tutores y corroborara su identidad cada vez que se pulsara el botón. Teniendo en cuenta esto y que esta funcionalidad no afectaba para nada al propósito del videojuego de cara a la fase de pruebas, no se llegó a implementar, dejando este botón a modo de *placeholder* (marcador de posición) para una posterior implementación. Esta nueva funcionalidad, una vez implementada, podría distribuirse como un parche o actualización del videojuego.

- **Componer las pistas de música de *CleanQuest*.** Como se comentó en el sub-apartado “Cambios significantes respecto a la preproducción” de la fase de producción (apartado 3.7.2), se decidió utilizar pistas de música de fondo con derechos de autor en el prototipo software de alto nivel de fidelidad. Se decidió así debido a que no se disponía experiencia para componer música propia y la música encontrada en repositorios gratuitos sin derechos de autor no encajaba dentro de la dirección artística propuesta para *CleanQuest*. Antes de poder comercializar el juego (aplicando el modelo de negocio planteado) sería necesario sustituir estas pistas por otras sin derechos de autor, componer unas nuevas u obtener la licencia para poder utilizar las actuales.

- **Aplicación de monitorización infantil para padres o tutores.** Otra posible mejora para *CleanQuest,* atendiendo a los productos de software analizados en el análisis del estado del arte (apartado 2) *Family Chores o Choremonster*, sería la creación de una aplicación hermana de *CleanQuest* de monitorización infantil para padres o tutores. Los productos de software mencionados, se dividían en un módulo para padres o tutores que permitía la monitorización del progreso de su hijo/a en el otro módulo, en este caso, la propia tabla de tareas domésticas. En *CleanQuest* se podría hacer algo parecido, creando una nueva aplicación de monitorización interconectada con *CleanQuest* y relegar en ella todo el control que tuvieran los padres de forma remota. De esta forma se relegaría en ella todo el proceso del control parental y a través de la misma los padres podrían comprobar que tareas

del “Desafío de realidad aumentada” han completado sus hijos, eliminando la necesidad de irlas comprobando físicamente una a una.

- **Creación de una comunidad de padres o tutores.** Muchos de los productos de software comentados en el análisis del estado del arte (apartado 2), cuentan con una web principal con foros o comunidades de usuarios. Estos foros normalmente están indicados para los padres o tutores de los hijos que usan los productos. De esta manera se consigue que los padres o tutores de los niños (usuarios) puedan comunicarse entre ellos, comentando aspectos sobre el producto y con la posibilidad de ponerse en contacto con los creadores del mismo o su servicio técnico directamente. Para *CleanQuest* se podría crear algo similar, aumentando su atractivo hacia los padres o tutores del usuario objetivo.

- **Creación de un estudio o empresa de cara a la distribución.** Con el plan distribución teórica de *CleanQuest* (modelo de negocio) establecido en el apartado de postproducción (3.8), convendría acompañarlo de la creación de algún tipo de pequeño estudio o empresa, de cara a la publicación. El que *CleanQuest* tenga por detrás un estudio o entidad similar, aportaría más confianza a los padres o tutores de los jugadores y llamaría más la atención a otras empresas del sector de videojuegos más grandes. Esto permitiría al supuesto estudio prosperar y mejorar *CleanQuest* en el futuro, incluso con la posibilidad de crear una franquicia con el mismo.

- **Interpretación de los vídeos de monitorización en las pruebas de validación de producto.** En el apartado de “Validación de producto” (apartado 4.4), durante las pruebas del videojuego realizadas con usuarios, varios padres de los hijos participantes enviaron vídeos de la experiencia de sus hijos mientras jugaban con *CleanQuest*. Como esto fue una iniciativa de los padres y no estaba dentro de la planificación para estas pruebas, no se tuvieron en cuenta en la interpretación de sus resultados. Aun así, después de su análisis, se pueden observar algunos patrones negativos en los niños a tener en cuenta. El más prevalente se trata de una pequeña frustración causada por los obstáculos en ciertas fases del mini-juego “Recoger los juguetes” (aunque más adelante se acabara por superarlos). Esto podría solucionarse simplemente reduciendo la dificultad en estas fases, pero igualmente se presentó la oportunidad de añadir una nueva característica al juego: Utilización de feedback positivo anti-frustración. Esto es, a partir de un número de intentos fallidos para conseguir un hito dentro de una fase en un mini-juego, aparecerá un mensaje de ánimo al jugador, por ejemplo “*Ánimo, casi lo consigues*”. Este mensaje podría transmitirse al jugador de forma textual (un *prompt*) o como un clip de audio. Esta nueva característica aumentaría aun más el atractivo del juego hacia el jugador objetivo, ya que este tipo *feedback* positivo es similar a los mensajes de ánimo que deberían recibir por parte de sus padres o tutores. Esta nueva funcionalidad, una vez implementada, podría distribuirse como un parche o actualización del videojuego.

6. Bibliografía

En este apartado se listan todas las referencias bibliográficas utilizadas a lo largo de esta memoria, incluyendo un enlace web al documento original si éste se encuentra disponible en internet para su consulta. Para documentar estas referencias se ha seguido el estándar ISO 690 tal y como recomienda la biblioteca de la universidad Carlos III [53]

6.1 Lista de referencias

[1] *Ametller Martínez, Lídia. "La obediencia de los hijos entre los 6 y los 12 años". Solo Hijos [web]. [Consultado el 03/07/14]. Disponible en:*
<http://www.solohijos.com/web/la-obediencia-de-los-hijos-entre-los-6-y-los-12-anos-2/>.

[2] *Rodríguez Menéndez, Mª del Carmen; Peña Clavo, José Vicente; Inda Caro, Mercedes. "La participación de los niños y niñas en las labores domésticas: Análisis discursivo de las opiniones parentales" [en línea]. Barcelona: Universitat de Barcelona, 2011. [Consultado el 03/07/14]. Disponible en:*
<http://www.cite2011.com/Comunicaciones/Familias/24.pdf>.

[3] *Klein, Wendy; P. Graesch Anthony; Izquierdo, Carolina. "Children and Chores: A Mixed-Methods Study of Children's Household Work in Los Angeles Families" [en línea]. California: American Anthropological Association, 2009. [Consultado el 03/07/14]. Disponible en:*
<http://celf.ucla.edu/2010_conference_articles/Klein_et_al_2009.pdf>.

[4] *de la Peña Palacios, Eva Mª. "Fórmulas para la igualdad nº 4. Reparto de tareas: Corresponsabilidad" [en línea]. Córdoba: Mancomunidad de Municipios Valle del Guadiato, 2007. [Consultado el 03/07/14]. Disponible en:*
<http://www.fundacionmujeres.es/maletincoeducacion/pdf/CUAD4horiz.pdf>.

[5] *Dishion, Thomas J.; Patterson, Scot G. (1996). "Preventive Parenting with Love, Encouragement, and Limits: The Preschool Years" [en línea]. EE.UU: Castalia Pub Co, enero de 1996. [Consultado el 03/07/14]. Disponible en:*
<http://dera.tempburbooks.eu/?id=preventive_parenting_with_love_encouragement_and_limits_the_preschool_years_thomas_j_dishion/>.

[6] *Webster-Stratton, Carolyn. "How to Promote Children's Social and Emotional Competence" [en línea]. Reino Unido: SAGE Publications, 1999. [Consultado el 03/07/14]. Disponible en:*
<http://books.google.es/books?hl=es&lr=&id=WX0i2WHSbi8C&oi=fnd&pg=PR8&dq=How+to+Promote+Children%27s+Social+and+Emotional+Competence&ots=VR0TZtwEwA&sig=FZ4z2X5aBJTESiFyS6T80A7KeBk#v=onepage&q&f=false>.

[7] *Springer, Matthew G. "Accountability Incentives. Do schools practice educational triage?" [en línea]. EE.UU: Vanderbilt University's Peabody College, noviembre de 2007. [Consultado el 03/07/14]. Disponible en:*
<http://media.hoover.org/sites/default/files/documents/ednext_20081_Springer_unabridged.pdf>.

[8] *Pérez-Barco, M.J. "Doce claves para que los hijos colaboren en las tareas domésticas". ABC [web]. 4 de febrero de 2013 [Consultado el 03/07/14]. Disponible en:*

<http://www.abc.es/familia-padres-hijos/20130204/abci-tareas-domesticas-ninos-201301241457.html>.

[9] *Millet, Eva. "Niños que ayudan en casa". La Vanguardia [web]. 11 de enero de 2013 [Consultado el 04/07/14]. Disponible en:*
<http://www.lavanguardia.com/estilos-de-vida/20130111/54358920062/ninos-que-ayudan-en-casa.html>.

[10] *Lee, Kang; Talwar, Victoria; McCarthy, Anjanie; Ross, Llana; Evans, Angela; Arruda, Cindy. "Can Classic Moral Stories Promote Honesty in Children?" [en línea]. EE.UU: Association for Psycological Science, 2014. [Consultado el 04/07/14]. Disponible en:*
<http://www.scilogs.com/next_regeneration/does-reading-moral-stories-to-children-promote-honesty/>.

[11] *Sacristan, Pedro Pablo. "Bedtime Stories, short stories with values". [Consultado el 04/07/14]. Disponible en:*
<http://freestoriesforkids.com/>.

[12] *Vallordo, Eric. "Las capacidades coordinativas". Entrenamiento deportivo [blog]. 1 de diciembre de 2008 [Consultado el 04/07/14].Disponible en:*
<http://entrenamientodeportivo.wordpress.com/2008/12/01/las-capacidades-coordinativas/>.

[13] *Granic, Isabela; Lobel, Adam; C. M. E. Engels, Rutger. [en línea]. EE.UU: American Psychological Association, enero de 2014. [Consultado el 04/07/14]. "The Benefits of Playing Video Games". Disponible en:*
<https://www.apa.org/pubs/journals/releases/amp-a0034857.pdf>.

[14] *Griffiths, Mark. "The educational benefits of videogames" [en línea]. Reino Unido: Nottingham Trent University, 2002. [Consultado el 04/07/14]. Disponible en:*
<http://sheu.org.uk/sites/sheu.org.uk/files/imagepicker/1/eh203mg.pdf>.

[15] *González-Bueno, Gabriel; Bello, Armando: Arias, Marta. "La infancia en España" [en línea]. Madrid: UNICEF España, 2012. [Consultado el 04/07/14]. Disponible en:*
<https://www.unicef.es/sites/www.unicef.es/files/Infancia_2012_2013_final.pdf>.

[16] *Evangelho, Jason. "2012's Massive Casual Gaming Footprint: Is The 'Hardcore' Audience Disappearing". Forbes [web]. 1 de febrero de 2013 [Consultado el 04/07/14]. Disponible en:*
<http://www.forbes.com/sites/jasonevangelho/2013/01/02/2012s-massive-casual-gaming-footprint-is-the-hardcore-audience-disappearing/>.

[17] ESA. *"The 2014 Essential Facts About the Computer and Video Game Industry" [en línea]. EE.UU: Entertaiment Software Association (ESA), abril de 2014. [Consultado el 04/07/14]. Disponible en:*

<http://www.theesa.com/wp-content/uploads/2014/10/ESA_EF_2014.pdf>.

[18] *"Video Game Development". Wikipedia [web]. [Consultado el 05/07/14]. Disponible en:*
<http://en.wikipedia.org/wiki/Video_game_development>.

[19] *"Agile Software Development". Wikipedia [web]. [Consultado el 05/07/14]. Disponible en:*
<http://en.wikipedia.org/wiki/Agile_software_development>.

[20] *"Adaptative software development". Wikipedia [web]. [Consultado el 05/07/14]. Disponible en:*
<http://en.wikipedia.org/wiki/Adaptive_software_development>.

[21] *"Iterative Design". Wikipedia [web]. [Consultado el 05/07/14]. Disponible en:*
<http://en.wikipedia.org/wiki/Iterative_design>.

[22] *"Waterfall Model". Wikipedia [web]. [Consultado el 05/07/14]. Disponible en:*
<http://en.wikipedia.org/wiki/Waterfall_model>.

[23] *"Jugabilidad". Wikipedia [web]. [Consultado el 05/07/14]. Disponible en:*
<http://es.wikipedia.org/wiki/Jugabilidad>.

[24] *IEEE Computer Society's; Technical Committee on Learning Technology. "Learning Technology, Volume 12 Issue 1" [en línea]. EE.UU: Technical Committee on Learning Technology (TCLT), enero de 2010. [Consultado el 05/07/14]. Disponible en:*
<http://www.ieeetclt.org/issues/january2010/index.htm>.

[25] *"Jugabilidad y experiencia del jugador". Wikispaces [web]. [Consultado el 05/07/14]. Disponible en:*
<http://jugabilidad.wikispaces.com/Producci%C3%B3n+y+Desarrollo+de+Videojuegos>.

[26] *"Top 15 Most Popular Search Engines". eBizMBA Guide [web]. Julio de 2014 [Consultado el 06/07/14]. Disponible en:*
<http://www.ebizmba.com/articles/search-engines>.

[27] *Forrest, Conner. "Apple v. Google: The goliath deathmatch by the numbers in 2014". Tech Republic [web]. 21 de marzo de 2014 [Consultado el 06/07/14]. Disponible en:*
<http://www.techrepublic.com/article/apple-v-google-the-goliath-deathmatch-by-the-numbers-in-2014/>.

[28] *Gagnon, Patricia. "Family Chores". [Consultado el 06/07/14]. Disponible en:*
<http://www.familychores.com/>.

[29] *Bergman, Chris; Armstrong, Paul. "Choremonster". [Consultado el 06/07/14]. Disponible en:*
<https://www.choremonster.com/>.

[30] *Ozone Development. "Abby's Home Laundry". Google Play [tienda web]. [Consultado el 06/07/14]. Disponible en:*
<https://play.google.com/store/apps/details?id=air.net.ozonedevelopment.AbbysHomeLaundry>.

[31] *Tabtale*. "Baby Home Adventure". *Google Play [tienda web]. [Consultado el 06/07/14].* Disponible en:
<https://play.google.com/store/apps/details?id=com.tabtale.babyhousechores&hl=es>.

[32] *Mobile Games Media. "Kids House Clean Up". Google Play [tienda web]. [Consultado el 06/07/14]. Disponible en:*
<https://play.google.com/store/apps/details?id=air.net.m7g7.KidsHouseCleanup>.

[33] *Toca Boca AB. "Toca House". iTunes [tienda web]. [Consultado el 06/07/14]. Disponible en:*
<https://itunes.apple.com/gb/app/toca-house/id495680460?mt=8>.

[34] *"Microtransaction". Wikipedia [web]. [Consultado el 06/07/14]. Disponible en:*
<http://en.wikipedia.org/wiki/Microtransaction>.

[35] "*Ley 34/1988, de 11 de noviembre, General de Publicidad*". Agencia Estatal Boletín Oficial del Estado [web]. *[Consultado el 06/07/14]. Disponible en:*
<http://www.boe.es/diario_boe/txt.php?id=BOE-A-1988-26156>.

[36] *Lee, Aaron. "15 essential mobile game development tools". Develop-online [web]. 15 de octubre de 2013 [Consultado el 09/07/14]. Disponible en:*
<http://www.develop-online.net/tools-and-tech/15-essential-mobile-game-development-tools/0184480>.

[37] *Reynolds, Christopher. "Top IOS & Android Mobile Game Development Tools, Frameworks, Engines & Resources". Moby Affiliates [blog]. 22 de octubre de 2012 [Consultado el 09/07/14]. Disponible en:*
<http://www.mobyaffiliates.com/blog/ios-android-mobile-game-development-tools-frameworks-engines-resources/>.

[38] *"All Android Game Engines". Mobile Game Engines [web]. [Consultado el 09/07/14]. Disponible en:*
<http://mobilegameengines.com/android/game_engines>.

[39] *Conder, Shane; Darcey, Lauren. "The Perfect Platform for Game Developers: Android". Developer [web]. 14 de mayo de 2014 [Consultado el 09/07/14]. Disponible en:*

<*http://www.developer.com/ws/android/client/the-perfect-platform-for-game-developers-android.html*>.

[40] *Viswanathan, Priya. "Best Tools for Mobile Game App Development". About Tech [web]. 17 de diciembre de 2013 [Consultado el 09/07/14]. Disponible en:*
<*http://mobiledevices.about.com/od/additionalresources/tp/Best-Tools-for-Mobile-Game-App-Development.htm*>.

[41] *Unity Technologies. Unity3d [web]. [Consultado el 10/07/14]. Disponible en:*
<*https://store.unity3d.com/es*>.

[42] *"100 Most Popular Engines Today". ModDb [web]. [Consultado el 10/07/14]. Disponible en:*
<*http://www.moddb.com/engines/top*>.

[43] *"Dashboards". Android Developers [web]. [Consultado el 10/07/14] Disponible en:*
<*https://developer.android.com/about/dashboards/index.html*>.

[44] *Dell'Antonia, KJ. "Age-Appropiate Chores for Children (and Why They're Not Doing Them)". NY Times [blog]. 24 de enero de 2014 [Consultado el 03/07/14.]. Disponible en:*
<*http://parenting.blogs.nytimes.com/2014/01/27/age-appropriate-chores-for-children-and-why-theyre-not-doing-them/?_r=0*>.

[45] *Child, Ben. "Disney turns away from hand-drawn animation". The Guardian [web]. 7 de marzo de 2013 [Consultado el 12/07/14]. Disponible en:*
<*http://www.theguardian.com/film/2013/mar/07/disney-hand-drawn-animation*>.

[46] "*Ley de Propiedad Intelectual* (*Real Decreto Legislativo 1/1996*)". Agencia Estatal Boletín Oficial del Estado [web]. *[Consultado el 06/07/14]. Disponible en:*
<http://www.boe.es/diario_boe/txt.php?id=BOE-A-1996-8930>.

[47] *Portal de administración electrónica del gobierno de España. "Métrica v.3" [en línea]. España: Gobierno de España. [Consultado el 12/07/14]. Disponible en:*
<*http://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html#.VIRc7zGG98E*>.

[48] *"Google Play". Wikipedia [web]. [Consultado el 12/07/14]. Disponible en:*
<*http://en.wikipedia.org/wiki/Google_Play*>.

[49] *Universidad de Sevilla. "Técnicas de evaluación dinámica" [en línea]. Sevilla: Universidad de Sevilla. [Consultado el 12/07/14]. Disponible en:*
<http://www.lsi.us.es/docencia/get.php?id=361>.

[50] *"Artículo 18 de la constitución española". Congreso de los diputados [web]. [Consultado el 15/07/14]. Disponible en:*

<*http://www.congreso.es/consti/constitucion/indice/titulos/articulos.jsp?ini=18&tipo=2*>.

[51] "*Ley 1/1982 de 5 de mayo referida al derecho al honor, a la intimidad personal y a la propia imagen*". Agencia Estatal Boletín Oficial del Estado [web]. *[Consultado el 15/07/14]. Disponible en:*
<http://www.boe.es/buscar/doc.php?id=BOE-A-1982-11196>.

[52] "*Ley 5/1999 de 13 de diciembre sobre la Protección de Datos de Carácter Personal (LOPD)*". Agencia Estatal Boletín Oficial del Estado [web]. *[Consultado el 15/07/14]. Disponible en:*
<http://www.boe.es/buscar/doc.php?id=BOE-A-1999-23750>.

[53] "*Cómo citar bibliografía: UNE-ISO 690*". Universidad Carlos III de Madrid [web]. *[Consultado el 17/07/14]. Disponible en:*
<http://portal.uc3m.es/portal/page/portal/biblioteca/aprende_usar/como_citar_bibliografia#recursos>.

Apéndices

Presupuesto del trabajo de fin de grado

Con el proceso de desarrollo del TFG concluido, es posible calcular los costes totales del mismo. Para ello, por un lado se realiza un balance o cálculo de los costes de los **recursos humanos** utilizados, esto es, los costes asociados a las horas desempeñadas en la realización de cada tarea en el TFG, por persona involucrada en el mismo. Por otro lado, se realiza otro cálculo de los costes asociados a la **amortización de los equipos y software informático** involucrados en el desarrollo. El presupuesto total estimado será equivalente al total de ambos costes.

Coste de los recursos humanos

Para calcular el coste de los recursos humanos, primeramente es necesario establecer los costes por hora de cada tarea realizada. Para ello se parte de lo que suele cobrar un ingeniero/programador junior por hora, 30,00 €, y se adapta según el esfuerzo estimado desempeñado en cada tarea en comparación con las demás. Esto es debido a la dificultad de formalizar un coste por hora base para las tareas más especializadas, como por ejemplo el modelado 3d, usualmente realizado por *freelances* en el ámbito profesional, con precios muy dispares). Teniendo esto en cuenta se proponen las siguientes tareas desempeñadas, junto con sus costes por hora:

- Investigación del problema: 30,00 €/hora
- Concepción y diseño artístico y conceptual: 50,00 €/hora
- Especificación de requisitos y diseño de implementación y técnico: 50,00 €/hora
- Modelado y animación 3d: 50,00 €/hora
- Modelado y composición 2d: 40,00 €/hora
- Sonido: 30,00 €/hora
- Implementación en *Unity 3d*: 30,00 €/hora
- Pruebas: 40,00 €/hora
- Redacción y maquetación de memoria: 30,00 €/hora
- Tutelaje y asesoramiento: 30,00 €/hora

Una vez establecidos los costes de cada tarea es necesario determinar las personas que han participado en el proceso de desarrollo del TFG, incluyendo a su autor y a las personas que colaboraron en mayor o menor medida en el mismo. En este caso se podría considerar al único miembro del equipo de desarrollo y autor de este TFG, Pablo Sánchez Ordaz; su tutor externo, Jorge Ruiz Magaña; su tutor interno de la universidad, M. Carmen Fernández Panadero; dos alumnos de la universidad, colaboradores menores en el trabajo, Jorge Lucena Rodríguez y Diego Trompeta Urretavizcaya; y un familiar del autor de TFG, Arturo Ordaz Muñoz que ayudó mediante el doblaje de los clips de voz para el juego. Cabe destacar que, debido a que algunas de estas personas, presentan una categoría profesional mayor (Ingeniero Senior), se considerará su desempeño más costoso de cara al desarrollo, estimándose el doble de coste por hora establecido para cada tarea.

A continuación se muestra el desglose y cálculo del coste total de los recursos humanos, reflejando cada persona y tarea pertinentes, en forma de tabla:

Persona	Tarea	Coste/hora	Horas empeñadas	Coste total
Pablo Sánchez Ordaz (Autor)	Investigación del problema	30,00 €	60 horas	1.800,00 €
	Concepción y diseño artístico y conceptual	50,00 €	145 horas	7.250,00 €
	Especificación de requisitos y diseño de implementación y técnico	50,00 €	100 horas	5.000,00 €
	Modelado y animación 3d	50,00 €	30 horas	1.500,00 €
	Modelado y composición 2d	40,00 €	12 horas	480,00 €
	Sonido	30,00 €	5 horas	150,00 €
	Implementación en *Unity 3d*	30,00 €	210 horas	6.300,00 €
	Pruebas	40,00 €	60 horas	2.400,00 €
	Redacción y maquetación de memoria	30,00 €	150 horas	4.500,00 €
Jorge Ruiz Magaña	Tutelaje y asesoramiento	60,00 €*	15 horas	900,00 €
M. Carmen Fernández Panadero	Tutelaje y asesoramiento	60,00 €*	10 horas	600,00 €
Jorge Lucena Rodríguez	Tutelaje y asesoramiento	30,00 €	5 horas	150,00 €
Diego Trompeta Urretavizcaya	Tutelaje y asesoramiento	30,00 €	3 horas	90,00 €
Arturo Ordaz Muñoz	Sonido	60,00 €**	4 horas	240,00 €
*Ingeniero Senior **Técnico superior de imagen y sonido		**Total**	**789 horas**	**31.360,00 €**

Tabla A1. Cálculo de los costes de los recursos humanos asociados al TFG

De esta forma el coste total estimado de los recursos humanos que han participado en el proceso de desarrollo del TFG asciende a 31.360,00 €

Coste de amortización de equipos y software informático

Para calcular el coste de la amortización de los equipos informáticos y software utilizados en el desarrollo de este TFG, se parte de la fórmula simplificada del cálculo de la amortización de un equipo informático:

*(Coste inicial / Vida útil) * Tiempo de utilización*

Debido a que durante el *tiempo de utilización* de los equipos y software informáticos, éstos no han sido utilizados enteramente al desarrollo del TFG, se adaptará la fórmula para incluir su factor de utilización referente al proyecto dentro del tiempo de utilización general. De esta forma la fórmula resultante queda de esta manera:

*(Coste inicial / Vida útil) * (Tiempo de utilización * Factor de utilización)*

Siendo el factor de utilización el porcentaje de utilización en el TFG estimado (0-1). La vida útil por su parte se estimará en 5 años para los ordenadores de sobremesa y portátiles, 3 años para los dispositivos móviles y 2 años para el software.

Teniendo esto en cuenta a continuación se muestra el desglose y cálculo del coste total de la amortización de software informático, reflejando los datos de la fórmula para su cálculo:

HW/SW	Coste inicial (€)	Vida útil (meses)	Tiempo de utilización (meses)	Factor de utilización (0-1)	Amortización (€)
Ordenador de sobremesa *Intel Core 2 Quad Q6600 @ 2.40GHz*	600,00	60	2	0.5	10,00
Ordenador portátil *ASUS Notebook N61Jq Series Intel Core i7 720QM @ 1.60GHz*	1000,00	60	4	0.9	60,00
Teléfono móvil (*Smartphone*) *Sony Xperia Neo V MT11i*	150,00	36	2	0.4	3,33
Teléfono móvil (*Smartphone*) *Sony Xperia Z1 C6903*	250,00	36	2	0.4	5,56
Teléfono móvil (*Smartphone*) *Huawei P6-U06*	100,00	36	1	0.2	0,56
Cámara de video digital *Samsung PL21*	600,00	36	1	0.1	1,67
Sistema Operativo *Windows 7 Ultimate 64 bits*	0,00	-	-	-	0,00
Sistema Operativo *Windows 7 Home Premium 64 bits SP1*	0,00	-	-	-	0,00
Sistema Operativo *Android 2.3.4 Gingerbread*	0,00	-	-	-	0,00
Sistema Operativo *Android 4.4.4 Kit Kat*	0,00	-	-	-	0,00

Sistema Operativo *Android 4.2.2 Jelly Bean*	0,00	-	-	-	0,00
Kit de desarrollo de videojuegos *Unity 4* Versión *4.5.0f6*	0,00	-	-	-	0,00
Programa de modelado 3d *Sculptris Alpha 6*	0,00	-	-	-	0,00
Editor de gráficos rasterizados *Adobe Photoshop CS5 Extended* Versión *12.0*	120,00	24	3	0.7	10,50
Editor de efectos visuales y *composición Adobe After Effects CS5* Versión *10.0.0.458*	120,00	24	1	0.2	1,00
Editor de audio *Audacity 2.0.3*	0,00	-	-	-	0,00
Capturador de pantalla *Fraps 3.5.99*	0,00	-	-	-	0,00
				Total	**92,62 €**

Tabla A2. Cálculo de los costes de la amortización de los equipos y software informático asociados al TFG

Coste de amortización de equipos y software informático

Con los dos tipos de costes calculados, el presupuesto total del proyecto será equivalente al total de ambos, esto es: 31.360,00 €+ 92,62 € = **31.452,62 €.**

Manual de usuario de *CleanQuest*

Gracias por descargar el videojuego *CleanQuest*. Antes de nada, cabe mencionar que este manual de usuario es prescindible, ya que el videojuego ha sido diseñado para ser totalmente intuitivo, sin necesidad de manuales o tutoriales de ningún tipo. Aun así, este manual de usuario resolverá todas sus dudas relacionadas con el uso de *CleanQuest*.

Contenidos

Introducción

CleanQuest es un videojuego educativo para dispositivos *Android*, recomendado para niños de entre 6 y 9 años.

En CleanQuest, el jugador tomar el papel de Jaimito, el hijo más pequeño de una el hijo más pequeño en una familia que vive en una casa rural con granja. El jugador, como Jaimito, deberá ayudar a distintos miembros de su familia a realizar tareas de limpieza de su casa en forma de divertidos mini-juegos; en el videojuego base: "Recoger los juguetes", "Barrer el suelo" y "Limpiar los cristales".

Cada vez que el jugador complete una fase de un mini-juego, se le otorga un premio en forma de cromo virtual coleccionable, de una rareza equivalente a la habilidad que haya demostrado en el mini-juego. El jugador podrá disfrutar de la colección de cromos que haya ido obteniendo en la sección "Galería de cromos" dentro del juego.

Así mismo, en ciertos momentos, el videojuego se dirigirá al propio jugador para invitarle a participar en el "Desafío de realidad aumentada". En este desafío el jugador deberá realizar las mismas tareas de limpieza del videojuego pero en la vida real, con el fin de obtener nuevos y raros cromos virtuales para añadir a la colección. La realización de las tareas en el desafío deberá ser confirmada por los padres o tutores del jugador, introduciendo un código específico en la sección "Galería de cromos", para así desbloquear los nuevos cromos como premio. Los códigos respectivos de cada tarea deberán ser establecidos la primera vez que se inicie el juego, por medio de un control parental.

Debido a sus características, **el uso de *CleanQuest* fomenta en sus jugadores la colaboración en las tareas de limpieza domésticas y mediante el "Desafío de realidad aumentada" se garantiza la realización de las mismas.**

Inicio de la aplicación

Antes de iniciar la aplicación, asegúrese de que se encuentra instalada en su dispositivo *Android*. Si ya cuenta con la aplicación instalada podrá iniciarla pulsando en el icono de la aplicación "*CleanQuest*" correspondiente.

Si es la primera vez que se inicia la aplicación, por motivos que se explicará más adelante en la sección de "Control parental", es recomendable que la inicien los padres o tutores del jugador. En este caso la aplicación se iniciará en el "Control parental".

Una vez que se haya completado el control parental, las subsiguientes veces que se inicie la aplicación, ésta se iniciará en el "Menú principal".

Figura A1. Icono de la aplicación CleanQuest

Control parental

El control parental aparece la primera vez que se inicia el juego. Como su propio nombre indica, esta parte del juego está indicada para ser completada **por los padres o tutores del jugador**. Su propósito es, por un lado, introducir a los padres o tutores del jugador a *CleanQuest* y por otro permitirles establecer los códigos de cada tarea del "Desafío de realidad aumentada" y, opcionalmente, el lugar del entorno doméstico donde deseen que las realice el jugador. La funcionalidad y el propósito de estos códigos y del "Desafío de realidad aumentada" en general serán comentados en detalle más adelante en la sección "Desafío de realidad aumentada" de este manual.

El control parental está compuesto por 3 pantallas, navegables mediante los botones que presentan en ambos laterales inferiores. La primera pantalla es de bienvenida.

La segunda pantalla muestra un vídeo introductorio sobre el videojuego, donde se describen sus características generales y se explica el propósito del control parental, junto con indicaciones de cómo completarlo. Si no puede visualizar éste video correctamente en su dispositivo *Android*, también se encuentra disponible en el siguiente enlace: <https://www.youtube.com/watch?v=RwZ_BcwBD00>. Una vez visualizado el vídeo puede volverlo a visualizar si lo desea pulsando el botón con el símbolo de reproducción central de la pantalla.

Figura A2. 1ª y 2ª pantallas del "Control parental"

En la tercera y última pantalla se muestran 3 botones verticales con imágenes que representan cada tarea de limpieza que realizará el jugador en el "Desafío de realidad aumentada". Al pulsar cada una de ellas aparecerá una sub-pantalla mediante la cual se podrá establecer el código específico para esa tarea y opcionalmente, el lugar de

realización deseado para la misma. Por ejemplo, para establecer el código y el lugar para la tarea “Barrer el suelo”, primeramente se pulsa en el botón con la imagen que representa la tarea (el niño barriendo). A continuación se pulsa en los respectivos botones de entrada de texto y se establece como código “Fresa” y como lugar de realización “Tu habitación”, mediante el teclado táctil nativo del dispositivo (figura A3).

Figura A3. Ejemplo de introducción de contraseña y lugar en la 3ª pantalla del Control parental

Una vez que se han introducido ambos datos, para cerrar la sub-pantalla y volver a la 3ª pantalla del control parental pulse sobre el botón con el símbolo “X”. Si el código se ha introducido correctamente, el botón que representa la tarea debería rodearse por un halo verdoso. En caso contrario, si el código no se ha introducido correctamente (ej. Se ha dejado en blanco), el botón se rodeará con un halo de color rojo. El botón para avanzar en esta pantalla y dar por completado el control parental, se encontrará bloqueado (translucido) hasta que se hayan introducido todos los códigos correctamente y los 3 botones se encuentren rodeados con un halo verdoso. Al pulsar éste botón se procederá al “Menú principal”.

Una vez completado el control parental, los padres o tutores del jugador podrán entregar el dispositivo *Android* al jugador para que pueda disfrutar de *CleanQuest*.

Figura A4. 3ª pantalla del Control parental con el botón de avance desbloqueado

Menú principal

El menú principal es la pantalla principal del juego, desde la que se puede acceder al resto de pantallas. Una vez completado el "Control parental", ésta será la pantalla que aparecerá por defecto cada vez que se vuelva a iniciar la aplicación.

Figura A5. Menú principal de CleanQuest

El botón "X" de la esquina superior izquierda, cierra la aplicación. El botón con una imagen de un álbum de cromos de la parte superior derecha, lleva a la "Galería de cromos". Los tres botones inferiores representan los tres mini-juegos de limpieza doméstica de los que se compone *CleanQuest*, de izquierda a derecha: "Recoger los juguetes", "Barrer el suelo" y "Limpiar los cristales". Al pulsar en alguno de ellos aparecerá la sub-pantalla de selección de fase para ese mini-juego en concreto.

Figura A6. Sub-pantalla de selección de fase del mini-juego "Barrer el suelo"

Desde esta sub-pantalla, al pulsar el botón correspondiente de alguna fase, procederá a la fase del mini-juego correspondiente. Inicialmente todas las fases excepto la primera aparecerán bloqueadas. Para desbloquearlas es necesario completar, al menos una vez, cada fase anterior a la que se pretende desbloquear. Debajo de cada uno de los botones de fases se muestra el trofeo de mayor valor conseguido durante esa fase en concreto (estos trofeos se comentarán más en detalle

en la sección “Mini-juegos” de este manual). Pulsando el botón “X” se cierra esta sub-pantalla y se vuelve al menú principal.

Galería de cromos

La galería de cromos puede ser accedida desde el menú principal. En ella se pueden visualizar los cromos virtuales coleccionables obtenidos como premios en los mini-juegos o desbloqueados al completar alguna tarea del “Desafío de realidad aumentada”. Igualmente la galería de cromos permite a los padres o tutores del jugador introducir los códigos de las tareas del “Desafío de realidad aumentada” (establecidas en el “Control parental”), para desbloquear premios en forma de nuevos cromos coleccionables tras confirmar la realización de éstas por el jugador.

Al entrar en la galería de cromos, aparecerá la pantalla de selección de sección de la galería.

Figura A7. Pantalla de selección de sección de la galería de cromos

El botón “X” de la esquina inferior izquierda, se utiliza para volver al “Menú principal”. El botón “Ver Cromos” abre la sub-pantalla de visor de cromos. El botón “Obtener cromos, desafío de realidad aumentada” abre la sub-pantalla de introducción de códigos para el “Desafío de realidad aumentada”.

Figura A8. Sub-pantalla de visor de cromos de la galería de cromos

Desde la sub-pantalla de visor de cromos, se pueden visualizar los cromos virtuales coleccionables obtenidos como premios en los mini-juegos o desbloqueados al completar alguna tarea del "Desafío de realidad aumentada", a modo de álbum de cromos. Al pulsar sobre algún cromo obtenido, aparecerá el modelo 3d del cromo y su nombre como muestra la figura A8. Los botones con flechas de los laterales inferiores sirven para navegar entre las páginas del álbum. El botón "Salir", sirve para volver a la pantalla de selección de sección de la galería.

Figura A9. Sub-pantalla de introducción de códigos para el "Desafío de realidad aumentada".

Desde la sub-pantalla de introducción de códigos para el "Desafío de realidad aumentada", al pulsar el botón "Introducir código", los padres o tutores del jugador pueden introducir los códigos de las tareas del "Desafío de realidad aumentada" (los que fueron establecidos en el "Control parental") para desbloquear nuevos cromos coleccionables. Al introducir un código de una tarea correctamente (uno que se corresponda con el código establecido para esa tarea), aparecerá un mensaje de aviso con el cromo correspondiente desbloqueado, como premio. En caso contrario, si el código introducido no se corresponde con ninguno de los establecidos en el "Control parental" aparecerá un mensaje de error. Los cromos desbloqueados de esta manera se añadirán a la galería de cromos inmediatamente.

Por supuesto, **es recomendable que los padres o tutores del jugador introduzcan estos códigos una vez que confirmen que el jugador ha completado realmente la tarea** correspondiente del "Desafío de realidad aumentada" en la vida real (ej. Ha barrido el suelo de su habitación), ya que en caso contrario se perdería el valor educativo y el sentido del "Desafío de realidad aumentada". El "Desafío de realidad aumentada" en general se comentará en detalle más adelante, en la sección de "Desafío de realidad aumentada".

El botón "Salir" de esta sub-pantalla, sirve para volver a la pantalla de selección de sección de la galería. El botón "Resetear control parental" no tiene funcionalidad en esta versión de *CleanQuest*.

Mini-juegos

CleanQuest se compone de 3 mini-juegos, los cuales representan tareas de limpieza doméstica clásicas: “Recoger los juguetes”, “Barrer el suelo” y “Limpiar los cristales”. Cada mini-juego se compone de 3 fases, que pueden ser accedidas desde el “Menú principal”.

Reglas comunes de los mini-juegos

En estos mini-juegos el jugador, en la piel de Jaimito, deberá ayudar a sus familiares a realizar estas tareas de limpieza domésticas. Al empezar una fase de un mini-juego, el familiar al que Jaimito tenga que ayudar en ese mini-juego, mantendrá una conversación con él y le explicará el motivo por el cual tiene que ayudarle. Para finalizar la conversación y comenzar a jugar la fase pulse el botón “Empezar”.

Figura A10. Pantalla de inicio de fase del mini-juego “Recoger los juguetes”

Aunque los mini-juegos de *CleanQuest* presentan una jugabilidad distinta unos de otros, éstos se rigen por unas reglas y objetivo comunes: En cada fase, el jugador deberá completar una serie de hitos en el menor tiempo posible (contador progresivo). Por ejemplo, en el mini-juego “Recoger los juguetes”, se consigue completar un hito metiendo un juguete en el baúl de los juguetes.

Figura A11. Muestra del mini-juego “Recoger los juguetes” con marcadores en pantalla

Para llevar la cuenta de los hitos que se van consiguiendo en una fase y los hitos totales restantes, así como el tiempo transcurrido en segundos, en la fase aparecen unos marcadores de tiempo e hitos en la parte superior de la pantalla, como se muestra en la figura A11. En el marcador de tiempo igualmente se muestra el trofeo actual al que opta el jugador en ese rango de tiempo, y el tiempo restante para optar al siguiente trofeo de menor valor. Los trofeos se comentarán más adelante en la "Dinámica de premios".

Como jugar al mini-juego "Recoger los juguetes"

En las fases de éste mini-juego el jugador deberá recoger los juguetes de la habitación de Jaimito e introducirlos en el baúl de los juguetes que aparece en pantalla. Los juguetes por su parte pueden ser identificados por un halo amarillo que les rodea.

Cada vez que un juguete se introduzca en el baúl, se contará como un hito y el marcador de hitos ("Juguetes" en este caso) se incrementará en 1. Para recoger un juguete el jugador deberá pulsar sobre él en la pantalla táctil, arrastrarlo sin dejar de pulsar hasta el interior del baúl de los juguetes y soltarlo para introducirlo.

Figura A12. Muestra del mini-juego "Recoger los juguetes"

Como jugar al mini-juego "Barrer el suelo"

En las fases de éste mini-juego el jugador deberá barrer con una escoba las pelusas del salón de Jaimito hasta depositarlos en el rincón con una línea amarilla que aparece en pantalla. Las pelusas por su parte pueden ser identificados pequeñas bolas rodeadas por un halo gris.

Cada vez que una pelusa atraviese la línea amarilla del rincón indicado, se contará como un hito y el marcador de hitos ("Pelusas" en este caso) se incrementará en 1. Para manejar la escoba y poder barrer las pelusas el jugador puede hacer uso del *joystick* táctil del lado inferior izquierdo de la pantalla y del set de botones del lado inferior derecho. Mediante el *joystick* se puede desplazar la escoba de un lado a otro de la habitación, barriendo las pelusas que se encuentren con ella. El botón superior del set de botones (el más pequeño de los dos), permite girar 90º la escoba. El botón

inferior del set de botones (el más grande de los dos) permite agitar la escoba bruscamente (dar un “escobazo”), para desplazar las pelusas cercanas a la escoba con más velocidad hacia donde apunte ésta.

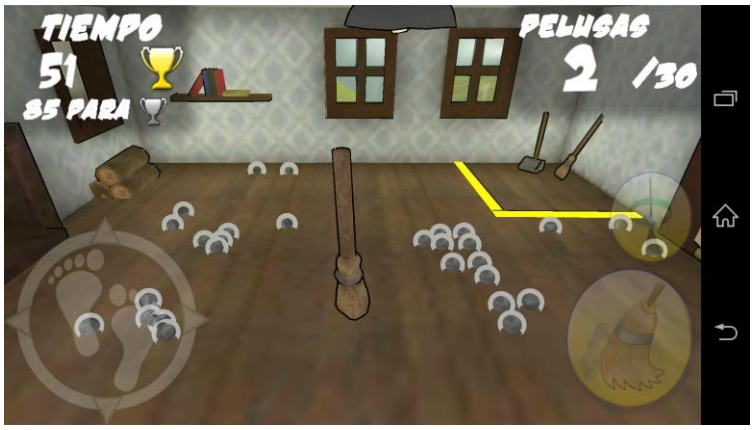


Figura A13. Muestra del mini-juego “Barrer el suelo”

Como jugar al mini-juego “Limpiar los cristales”

En las fases de éste mini-juego el jugador deberá limpiar con un trapo las manchas de barro de una cristalera de la granja de Jaimito hasta dejarla libre de manchas.

Cada vez que se limpie una mancha de barro hasta hacerla desaparecer, se contará como un hito y el marcador de hitos (“Manchas” en este caso) se incrementará en 1. Para limpiar una mancha de barro, el jugador debe pulsar en el trapo y, sin dejar de pulsarlo, restregarlo de un lado a otro sobre las manchas que quiera limpiar. Cada vez que el jugador pase el trapo sobre una mancha, ésta se reducirá en tamaño, hasta que se desvanezca completamente.

Figura A14. Muestra del mini-juego “Limpiar los cristales”

Dinámica de premios

Al no existir límite de tiempo, el jugador completa una fase cuando consigue todos los hitos indicados en el marcador de hitos de esa fase. Al completarla, se presentará al jugador con otra conversación con el mismo familiar del comienzo de la fase. En esta conversación el familiar de Jaimito felicitará al jugador por su labor en el mini-juego y le mostrará los premios obtenidos en esa fase.

Figura A15. Pantalla de final de fase del mini-juego "Recoger los juguetes"

Los premios obtenidos pueden ser trofeos y cromos virtuales coleccionables asociados a los trofeos. Los trofeos presentan un valor equivalente al tiempo que haya tardado el jugador en completar una fase, siendo trofeo de oro, plata, bronce o juguete los posibles valores a obtener. Cuanto menos se tarde en completar una fase, se obtendrá un trofeo de mayor valor, siempre optando al trofeo de juguete como premio de menor valor. Al obtener por primera vez un trofeo de un valor determinado se desbloqueará a su vez el cromo virtual coleccionable asociado a ese trofeo en particular y todos los demás cromos asociados a trofeos de menor valor en esa fase. Por ejemplo, al completar una fase con un tiempo equivalente al trofeo de oro, como premios se obtendrán el trofeo de oro y los cromos virtuales coleccionables asociados a los trofeos de oro, plata, bronce y juguete. Los cromos obtenidos de esta manera aparecerán en la "Galería de cromos" inmediatamente.

En la conversación de final de fase, cuando el familiar esté presentando cada premio, para avanzar de un premio a otro, se pulsa el botón "Siguiente". Una vez que se hayan mostrado todos los premios, para finalizar la fase, se pulsa el botón "Terminar" que aparecerá. Al pulsar este botón, se procederá a la siguiente fase del mini-juego o, si es la tercera y última fase del mismo, se procederá a la presentación del desafío de realidad aumentada de la tarea que representa ese mini-juego concretamente. La presentación del desafío de realidad aumentada se comentará más en detalla en la sección "Desafío de realidad aumentada" de este manual.

Pausar y salir de una fase en un mini-juego

Para pausar una fase en la que esté jugando, sólo tiene que pulsar el botón "Atrás" ("Back") o "Menú" de su dispositivo *Android*. Al hacerlo aparecerá la sub-pantalla de juego pausado.

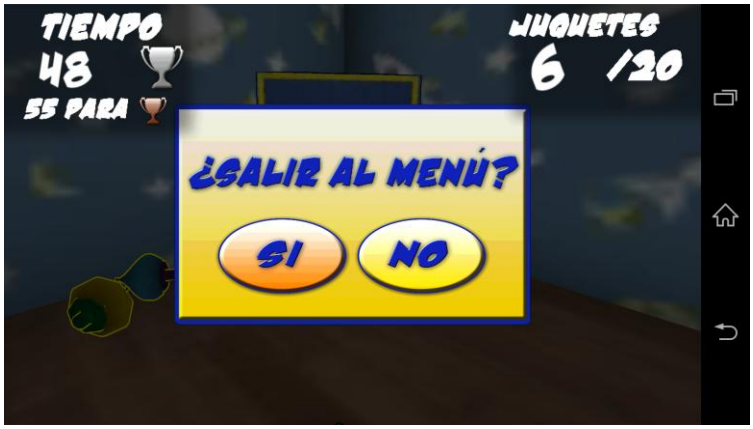


Figura A16. Muestra de la sub-pantalla de juego pausado

Desde esta sub-pantalla, al pulsar el botón "Si", saldrá del juego y volverá al "Menú principal". Pulsando el botón "No", reanudará el juego.

Desafío de realidad aumentada

Una vez completadas las 3 fases de un mini-juego, se mostrará al jugador la presentación del desafío de realidad aumentada. Aquí, el familiar de Jaimito que asistió en el inicio y el final de las fases del mini-juego mantendrá una conversación con el jugador, refiriéndose al mismo, no a Jaimito ("rompiendo el cuarto muro"). El familiar de Jaimito explicará al jugador en qué consiste el desafío de realidad aumentada respectivo a la tarea limpieza que representa la tarea del mini-juego que acaba de completar y le invitará a participar en él.

Figura A17. Presentación del desafío de realidad aumentada de la tarea "Recoger los juguetes"

El desafío de realidad aumentada de una tarea, consiste en que el jugador realice la tarea de limpieza en la vida real, en un lugar determinado. Este lugar es por defecto la casa del jugador, pero puede aparecer en la conversación como el establecido por los padres o tutores del jugador en el "Control parental" para esa tarea. En la conversación se indica al jugador que, una vez completado el desafío, le pida a sus padres o tutores que introduzcan en la "Galería de cromos" el código de esa tarea para desbloquear un nuevo y valioso cromo virtual coleccionable.

Los padres o tutores del jugador, por su parte, una vez que el jugador les pida introducir el código para el desafío de realidad aumentada, deberían confirmar que efectivamente el jugador ha realizado la tarea del desafío en la vida real, en el lugar determinado. Una vez confirmado, éstos podrán introducir el código respectivo de esa tarea (establecido previamente en el "Control parental") en la "Galería de cromos" y así desbloquear un nuevo y raro cromo para el jugador como premio por su esfuerzo.

Un ejemplo de realización de una tarea del desafío de realidad aumentada, continuando con el ejemplo de la figura A3, puede ser:

"*Raquel, una niña de 6 años de edad, se encuentra jugando a CleanQuest. Ella completa las 3 fases del mini-juego "Barrer el suelo" y el juego le muestra la presentación de desafío de realidad aumentada para la tarea análoga. En esta presentación, un familiar de Jaimito se dirige a ella y la explica que para superar el desafío de realidad aumentada, debe barrer el suelo en su habitación. Le indica que una vez que lo haga pida a sus padres o tutores el código para el desafío de realidad aumentada de la tarea "Barrer el suelo".*

Raquel, como es una buena chica, barre su habitación y le pide a su padre el código del desafío. Su padre, por su parte, comprueba que efectivamente Raquel ha barrido su habitación e introduce el código "Fresa" en la galería de cromos, desbloqueando así un nuevo cromo para la colección de Raquel."

Documento de autorización para la publicación de videos

A continuación se muestra el documento enviado a los padres participantes de las pruebas de validación de producto (apartado 4.4) que enviaron videos con la monitorización de sus hijos utilizando *CleanQuest*. El objetivo de este documento era confirmar su aprobación para la publicación de estos videos tanto en internet, como en esta memoria.

"Este documento tiene por objetivo reconocer la aprobación por parte de los padres o tutores para la publicación de los videos de monitorización enviados durante el transcurso de las pruebas de validación de producto del TFG (trabajo de fin de grado) "*CleanQuest. Simulador para fomentar la colaboración infantil en las tareas de limpieza del hogar*", realizado por el alumno Pablo Sánchez Ordaz de la Universidad Carlos III de Madrid. Estos videos serán publicados con fines académicos en el sitio web "*www.youtube.com*" así como en la memoria del propio TFG.

Esta autorización viene regulada por el artículo 18 de la Constitución Española, la Ley 1/1982 de 5 de mayo referida al *derecho al honor, a la intimidad personal y familiar y a la propia imagen*. Igualmente se acoge a la Ley 5/1999 de 13 de diciembre sobre la *Protección de Datos de Carácter Personal* (LOPD).

AUTORIZACIÓN

Don/Doña_____ con DNI _____ como padre / madre / tutor de _____ AUTORIZO la publicación de los videos con una finalidad estrictamente académica.

Firma del padre/madre/tutor Firma del hijo/a Firma del autor del TFG"